

# Insect survey 2020, basic analysis

*Jens Å*

*13 December, 2020*

## Contents

<b>Intro</b>	<b>2</b>
<b>Data import and cleaning</b>	<b>2</b>
Write a species list for the entomologists . . . . .	4
<b>Filter out the Nerlandsøya samples</b>	<b>6</b>
Check the emptying dates. . . . .	7
<b>Check missing weight data</b>	<b>11</b>
<b>Summaries of samples</b>	<b>12</b>
<b>Basic species richness graphs</b>	<b>13</b>
<b>Basic biomass graphs</b>	<b>15</b>
<b>Check overlap between malaise and window traps</b>	<b>18</b>
Total number of coleoptera species per trap type . . . . .	18
Percentage of catches in each trap type per species . . . . .	18
<b>Figs to Niclas on Pollinators</b>	<b>50</b>
<b>Phenology graphs</b>	<b>53</b>
<b>Plot some weather data</b>	<b>57</b>
<b>Sampling time analysis</b>	<b>59</b>
A similar comparison, but for biomass. . . . .	71
<b>Compare the ethanol to the propglykol window traps</b>	<b>75</b>
<b>Ano-registrations</b>	<b>79</b>
<b>Forest-registrations</b>	<b>85</b>
<b>Glyphosate figure</b>	<b>91</b>

## Intro

This script performs some basic analyses and produces figures for a report on the activity of the first year of a Norwegian insect monitoring program, in 2020. The project is financed by the Environmental Agency in Norway, and is the first continuous, general insect monitoring program in Norway. We employ malaise and window traps in semi-natural and forest locations. The monitoring is currently delimited to Eastern Norway. Further information about the project can be found here:

<https://www.nina.no/V%C3%A5re-fagomr%C3%A5der/Milj%C3%B8overv%C3%A5king-p%C3%A5-land/Oversikt>

This is a live document that contains code in a pretty raw format. It's not intended to be reproducible at this point. A separate document holds code relating to the database, where the data is stored.

## Data import and cleaning

At this point I get the raw data from Marie, which requires some cleaning. In the future, we'll put this all in a database, and separate out the cleaning and database import in a separate script. This will then start with instructions on how to access the raw data from public sources, so that it is reproducible.

Mind that this is still draft.

```
dat <- read_delim("../Data/Genetikkdata/long_format_data_GCF-2020-735_736_743_744_748_233",
  delim= ";",
  na = c("NA",
        "N/A",
        "Maa veies",
        "Smaa flasker maa veies"),
  col_types = cols(
    kingdom = col_character(),
    phylum = col_character(),
    class = col_character(),
    order = col_character(),
    family = col_character(),
    genus = col_character(),
    species = col_character(),
    sample = col_character(),
    value = col_double(),
    GenlabID = col_character(),
    GUID = col_character(),
    mottatt_lab = col_character(),
    locality = col_character(),
    trap = col_character(),
```

```

date_placed = col_date(format = "%d.%m.%Y"),
time_placed = col_time(format = ""),
empty_week = col_double(),
dateemptied = col_date(format = "%d.%m.%Y"),
timeemptied = col_time(format = ""),
liquid = col_character(),
personemptied = col_character(),
sample_id = col_character(),
comment = col_character(),
Tilsatt_antall_C.analis_SI_USA_08.06.20 = col_double(),
Tilsatt_antall_C.chinensis_08.06.20 = col_double(),
Tilsatt_antall_Sirisser = col_double(),
Tilsatt_antall_Melormer = col_double(),
Tilsatt_antall_C.maculatus = col_double(),
Vekt_tom_flaske_.g. = col_double(),
Vekt_flaske_med_EtOH_og_insekter_.g. = col_double(),
Vekt_flaske_.insekter_vaat_.g. = col_double(),
Vekt_flaske_.insekter_toerr_.g. = col_double(),
Totalvolum_.mengde_ATL_.prot.K_mL. = col_double(),
Vaatvekt_.g. = col_double(),
Toerrvekt_.g. = col_double(),
Kommentar_proeve = col_character(),
Ekstraksjonsdato = col_date(format = "%d.%m.%Y"),
Ekstraksjonskit = col_character(),
Subsamplet_volum_.ul. = col_double(),
Elueringsvolum = col_double(),
OD = col_double(),
X260_280 = col_double(),
X260_230 = col_double(),
Labperson = col_character(),
Ekstraksjonskommentar = col_character(),
kommentar_genlab = col_character(),
run = col_character(),
taxonomic_level = col_character(),
identification_confidence = col_character(),
possible_marker_issue = col_logical(),
probable_database_insufficiency = col_logical(),
norsk = col_logical(),
antall_arter_i_norge = col_double(),
antall_norsk_arter_i_db = col_double(),
pct_db_coverage = col_double()
)
)

```

Fix some coding issues.

```

# dat %>%
#   select(genus, species, new_species) %>%
# slice(297, 1902)
#
#
# dat %>%
#   select(genus, species, new_species) %>%
# slice(3219, 3585, 4017, 11424, 13275, 15465, 21056, 24168, 25077, 31567, 32805, 48502, 498
#
#Ekstraksjonskommentar = gsub(pattern = "[\\\"]", "", Ekstraksjonskommentar),
dat <- dat %>%
  mutate(species_latin = species)

dat <- dat %>%
  mutate(species = gsub(" sp", "_sp", species))

dat <- dat %>%
  mutate(species = gsub("[0-9](_)(sp)", "\\\\[\\]", species))

dat <- dat %>%
  mutate(species = gsub("(.*)(_)(.*)", "\\3", species))

dat <- dat %>%
  mutate(locality = str_to_sentence(locality))

```

Set factor levels

```

dat <- dat %>%
  mutate(locality = factor(locality, levels = c(paste0("Semi-nat_", 1:10), paste0("Skog_", 1:10),
                                             trap_type = ifelse(grepl("MF", trap), "Malaise", "Vindu"),
                                             habitat_type = ifelse(grepl("Skog", locality), "Skog", "SN_Gressmark"),
                                             liquid = ifelse(grepl("eth85", liquid), "Ethanol", "PropG1_Ethanol"),
                                             weeks_sampled = ifelse(grepl("1", trap) | grepl("3", trap), 2, 4)
                                             )))

```

## Write a species list for the entomologists

```

dat %>%
  select(kingdom,
         phylum,
         class,
         order,
         family,

```

```

        genus,
        species) %>%
distinct() %>%
arrange(kingdom,
        phylum,
        class,
        order,
        family,
        genus,
        species) %>%
write_excel_csv(file = "out/species_list_2020-11-02.csv")

```

Add missing bottle weights. This is based on a sample of bottle dry weight measurements made after the regular weighings.

```

bottle_weights <- tbl(con,
                      in_schema("catch_data", "bottle_weights")) %>%
select(-id)

# bottle_weights %>%
#   filter(bottle_type == "MF" & year == 2000) %>%
#   select(dry_weight) %>%
#   ggplot(.) +
#   geom_density(aes(x = dry_weight))

bottle_weights_MF <- bottle_weights %>%
  filter(bottle_type == "MF" & year == 2000) %>%
  select(dry_weight) %>%
  summarize(mean_weight = mean(dry_weight)) %>%
  collect()

## Warning: Missing values are always removed in SQL.
## Use `mean(x, na.rm = TRUE)` to silence this warning
## This warning is displayed only once per session.

bottle_weights_VF_stor <- bottle_weights %>%
  filter(bottle_type == "VF_stor" & year == 2000) %>%
  select(dry_weight) %>%
  summarize(mean_weight = mean(dry_weight)) %>%
  collect()

bottle_weights_VF_liten <- bottle_weights %>%
  filter(bottle_type == "VF_liten" & year == 2000) %>%
  select(dry_weight) %>%
  summarize(mean_weight = mean(dry_weight)) %>%
  collect()

```

Add mean bottle weights for the missing MF bottles.

```
sum(is.na(dat$Vekt_tom_flaske_.g.))

## [1] 7030

dat <- dat %>%
  mutate(Vekt_tom_flaske_.g. = ifelse(trap_type == "Malaise" & is.na(Vekt_tom_flaske_.g.), 

sum(is.na(dat$Vekt_tom_flaske_.g.))

## [1] 1874
```

The window traps is a little bit trickier, since we might use different bottle sizes. We have to add the big bottle weights to the samples that plausibly used big bottles, i.e. dry weights around the big bottle weights. Here we use a cutoff for the bottle types of 50 grams.

```
dat <- dat %>%
  mutate(Vekt_tom_flaske_.g. = ifelse(trap_type == "Vindu" & is.na(Vekt_tom_flaske_.g.) & V

sum(is.na(dat$Vekt_tom_flaske_.g.))
```

```
## [1] 590
```

Add the small bottle weights.

```
dat <- dat %>%
  mutate(Vekt_tom_flaske_.g. = ifelse(trap_type == "Vindu" & is.na(Vekt_tom_flaske_.g.) & V

sum(is.na(dat$Vekt_tom_flaske_.g.))

## [1] 0

dat <- dat %>%
  mutate(Vaatvekt_.g. = Vekt_flaske_.insekter_vaat_.g. - Vekt_tom_flaske_.g.,
         Toerrvekt_.g. = Vekt_flaske_.insekter_toerr_.g. - Vekt_tom_flaske_.g.)
```

## Filter out the Nerlandsøya samples

This is part of a cooperation with Norsøk, where we analysed the samples from one malaisetrap.

```
nerland_data <- dat %>%
  filter(locality == "Nerlandsoeya") %>%
  droplevels()

nerland_data %>%
  summarise(no_samples = n_distinct(sample))

## # A tibble: 1 x 1
```

```

##   no_samples
##             <int>
## 1          2
nerland_data %>%
  select(sample) %>%
  distinct()

## # A tibble: 2 x 1
##   sample
##   <chr>
## 1 OVERV_20_050_BF3BR2
## 2 OVERV_20_051_BF3BR2
nerland_data %>%
  summarise(no_spec = n_distinct(order, family, genus, species))

## # A tibble: 1 x 1
##   no_spec
##   <int>
## 1      513
write_csv(nerland_data,
           file = "out/Nerlandsoya_insekt_juli.csv")

dat <- dat %>%
  filter(locality != "Nerlandsoeya") %>%
  droplevels()

```

## Check the emptying dates.

The dates seems to be OK this time. Should do a proper check on the endpoints though.

```

dat <- dat %>%
  mutate(days_collected = as.numeric(dat$date_emptied - dat$date_placed))

# dat %>%
#   filter(days_collected == 0) %>%
#   select(sample_id,
#         GUID,
#         date_placed,
#         date_emptied) %>%
#   distinct() %>%
#   print(n = Inf)
#
# hist(as.numeric(dat$date_emptied - dat$date_placed))

```

```

dat %>%
  group_by(days_collected) %>%
  summarise(n())

## `summarise()` ungrouping output (override with `groups` argument)

## # A tibble: 20 x 2
##   days_collected `n()`
##   <dbl>     <int>
## 1 -6559      623
## 2 -6547      569
## 3 -15        3237
## 4 0          794
## 5 10         361
## 6 11         1065
## 7 12         6961
## 8 13         7286
## 9 14         18206
## 10 15         9000
## 11 16        14831
## 12 17         1411
## 13 25         558
## 14 26         1089
## 15 27         4247
## 16 28        19255
## 17 29         3584
## 18 30         435
## 19 31        1159
## 20 32         731

# dat %>%
#   filter(days_collected == 8) %>%
#   select(sample_id,
#         GUID,
#         date_placed,
#         date_emptied) %>%
#   distinct() %>%
#   print(n = Inf)

```

Fix a few typing errors in the dates. This is updated now in the sources, and should be fixed in the next data version.

```

dat %>%
  filter(days_collected <1) %>%
  select(locality, sample_id, trap, date_placed, date_emptied) %>%
  distinct()

```

```

## # A tibble: 13 x 5
##   locality    sample_id      trap date_placed date_emptied
##   <fct>      <chr>       <chr> <date>     <date>
## 1 Semi-nat_2 semi-nat_2_MF1_week_34 MF1  2020-08-19 2020-08-19
## 2 Semi-nat_2 semi-nat_2_MF1_week_36 MF1  2020-09-19 2020-09-04
## 3 Semi-nat_6 semi-nat_6_MF1_week_34 MF1  2020-08-03 2002-08-19
## 4 Semi-nat_6 semi-nat_6_MF2_week_34 MF2  2020-07-22 2002-08-19
## 5 Semi-nat_7 semi-nat_7_MF1_week_36 MF1  2020-09-19 2020-09-04
## 6 Semi-nat_8 semi-nat_8_MF1_week_36 MF1  2020-09-18 2020-09-03
## 7 Skog_10    skog_10_MF1_week_36  MF1  2020-09-18 2020-09-03
## 8 Skog_10    skog_10_VF1_week_36  VF1  2020-09-18 2020-09-03
## 9 Skog_10    skog_10_VF3_week_36  VF3  2020-09-18 2020-09-03
## 10 Skog_5    skog_5_MF1_week_36  MF1  2020-09-19 2020-09-04
## 11 Skog_5    skog_5_VF1_week_36  VF1  2020-09-19 2020-09-04
## 12 Skog_5    skog_5_VF3_week_36  VF3  2020-09-19 2020-09-04
## 13 Skog_8    skog_8_MF1_week_36  MF1  2020-09-18 2020-09-03

dat <- dat %>%
  mutate(date_placed = replace(date_placed, sample_id == "semi-nat_2_MF1_week_34", as.Date("2020-08-19")))
  mutate(date_placed = replace(date_placed, sample_id == "semi-nat_2_MF1_week_36", as.Date("2020-09-19")))
  mutate(date_emptied = replace(date_emptied, sample_id == "semi-nat_6_MF1_week_34", as.Date("2002-08-19")))
  mutate(date_emptied = replace(date_emptied, sample_id == "semi-nat_6_MF2_week_34", as.Date("2002-08-19")))
  mutate(date_placed = replace(date_placed, sample_id == "semi-nat_7_MF1_week_36", as.Date("2020-09-19")))
  mutate(date_placed = replace(date_placed, sample_id == "semi-nat_8_MF1_week_36", as.Date("2020-09-18")))
  mutate(date_placed = replace(date_placed, sample_id == "skog_10_MF1_week_36", as.Date("2020-09-18")))
  mutate(date_placed = replace(date_placed, sample_id == "skog_10_VF1_week_36", as.Date("2020-09-18")))
  mutate(date_placed = replace(date_placed, sample_id == "skog_10_VF3_week_36", as.Date("2020-09-18")))
  mutate(date_placed = replace(date_placed, sample_id == "skog_5_MF1_week_36", as.Date("2020-09-19")))
  mutate(date_placed = replace(date_placed, sample_id == "skog_5_VF1_week_36", as.Date("2020-09-19")))
  mutate(date_placed = replace(date_placed, sample_id == "skog_5_VF3_week_36", as.Date("2020-09-19")))
  mutate(date_placed = replace(date_placed, sample_id == "skog_8_MF1_week_36", as.Date("2020-09-18")))

dat <- dat %>%
  mutate(days_collected = as.numeric(dat$date_emptied - dat$date_placed))

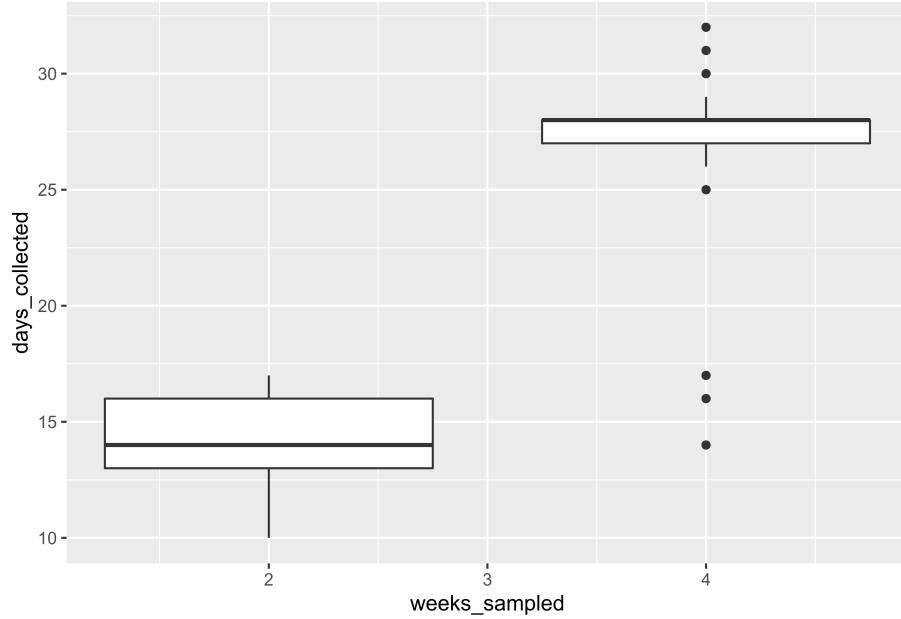
dat %>%
  filter(days_collected < 1) %>%
  select(locality, sample_id, trap, date_placed, date_emptied) %>%
  distinct()

## # A tibble: 0 x 5
## # ... with 5 variables: locality <fct>, sample_id <chr>, trap <chr>,
## #   date_placed <date>, date_emptied <date>

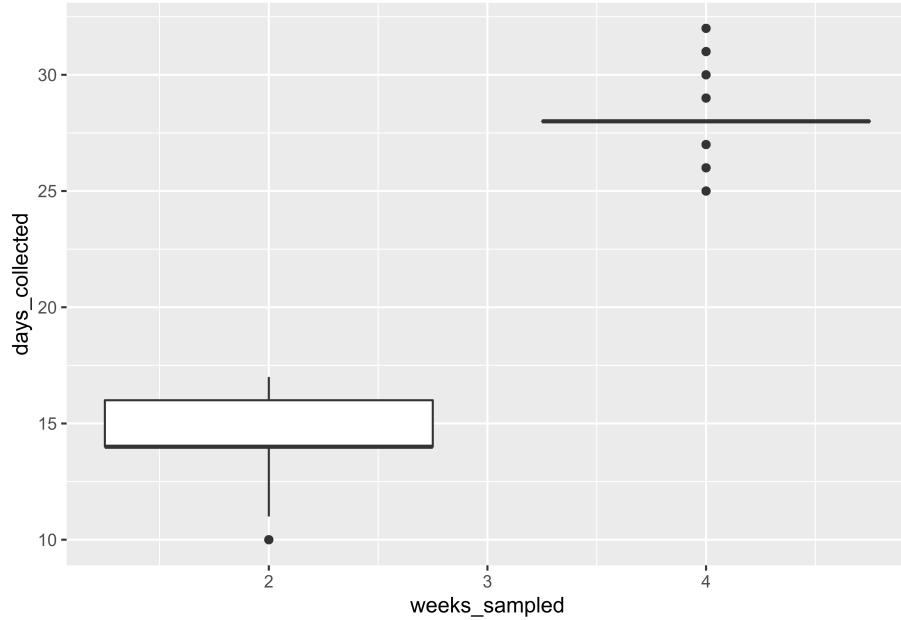
```

Set weeks sampled manually for the few locations in Oslo that got an extra short sampling in the beginning.

```
ggplot(dat, aes(x = weeks_sampled, y = days_collected, group = weeks_sampled)) +  
  geom_boxplot()
```



```
twoWeeks <- dat %>%  
  filter(weeks_sampled == 4 &  
         days_collected < 20) %>%  
  select(sample_id) %>%  
  distinct() %>%  
  as.vector()  
  
dat <- dat %>%  
  mutate(weeks_sampled = ifelse(sample_id %in% twoWeeks[[1]], 2, weeks_sampled))  
  
ggplot(dat, aes(x = weeks_sampled, y = days_collected, group = weeks_sampled)) +  
  geom_boxplot()
```



## Check missing weight data

```
dat %>%
  group_by(empty_week) %>%
  select(Vekt_tom_flaske_.g.,
         Vekt_flaske_med_EtOH_og_insekter_.g.,
         Vekt_flaske_.insekter_vaat_.g.,
         Vekt_flaske_.insekter_toerr_.g.,
         Vaatvekt_.g.,
         Toerrvekt_.g.
  ) %>%
  distinct() %>%
  summarise_all(~sum(is.na(.)))

## Adding missing grouping variables: `empty_week`
## # A tibble: 8 x 7
##   empty_week Vekt_tom_flaske~ Vekt_flaske_med~ Vekt_flaske_.i~ Vekt_flaske_.i~
##   <dbl>          <int>          <int>          <int>          <int>
## 1 22            0              0              0              0
## 2 24            0              0              0              0
## 3 26            0              3              0              2
## 4 28            0              9              0              0
```

```

## 5      30      0      2      0      2
## 6      32      0      0      0      0
## 7      34      0      0      0      0
## 8      36      0      0      0      0
## # ... with 2 more variables: Vaatvekt_.g. <int>, Toerrvekt_.g. <int>

```

## Summaries of samples

Total weight of insect catches.

```

dat %>%
  group_by(sample_id) %>%
  summarise(dry_weight = mean(Toerrvekt_.g., na.rm = T),
             wet_weight = mean(Vaatvekt_.g., na.rm = T)) %>%
  summarise(tot_dry_weight = sum(dry_weight, na.rm = T),
             tot_wet_weight = sum(wet_weight, na.rm = T))

## `summarise()` ungrouping output (override with `groups` argument)

## # A tibble: 1 x 2
##   tot_dry_weight tot_wet_weight
##       <dbl>        <dbl>
## 1     1924.       8325.

```

Total number of taxa

```

dat %>%
  summarise(no_taksa = n_distinct(order, family, genus, species))

## # A tibble: 1 x 1
##   no_taksa
##       <int>
## 1     11709

```

This include samples by 2020-09-04.

```

max(dat$dateemptied)

## [1] "2020-09-04"

```

How many of the taxa have we identified to species level?

```

not_id <- dat %>%
  filter(grepl("sp.", species)) %>%
  summarise(no_identified = n_distinct(order, family, genus, species))

not_id

## # A tibble: 1 x 1

```

```

##   no_identified
##                 <int>
## 1             6899
id <- dat %>%
  filter(!grepl("sp. ", species)) %>%
  summarise(no_identified = n_distinct(order, family, genus, species))

id

## # A tibble: 1 x 1
##   no_identified
##                 <int>
## 1             4810

```

We still lack species identifications on 58.9204885131096 percent of the taxa.

## Basic species richness graphs

Quick look at diversity at the different sites.

```

total_no_taxa <- dat %>% select(order, family, genus, species) %>%
  distinct() %>%
  count()

dat %>%
  filter(grepl("MF", trap)) %>%
  group_by(locality, habitat_type) %>%
  select(genus, species, habitat_type) %>%
  distinct() %>%
  count() %>%
  ungroup() %>%
  ggplot(.) +
  geom_bar(aes(y = n, x = locality, fill = habitat_type),
            stat = "identity") +
# ggttitle(paste0("Kumulativ artsrikedom i\nalle malaisefeller per ", max(dat$dateemptied)),
# theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
#       legend.position = "none") +
#       scale_fill_nina() +
#       ylab("Antall taksa") +
#       xlab("Lokalitet"))

## Adding missing grouping variables: `locality`
dat %>%
  filter(grepl("VF", trap)) %>%

```

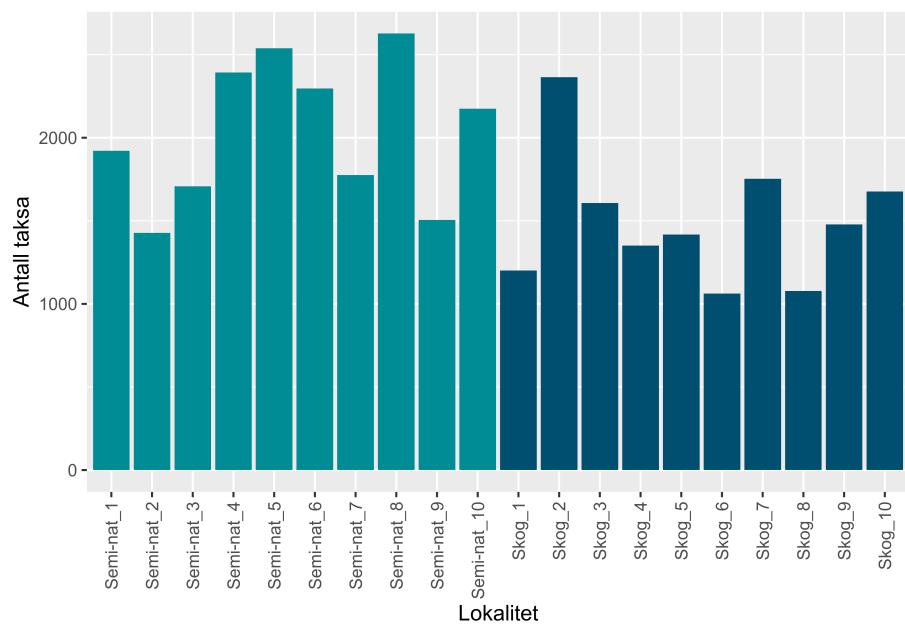


Figure 1: Cumulative number of taxa found in malaise traps.

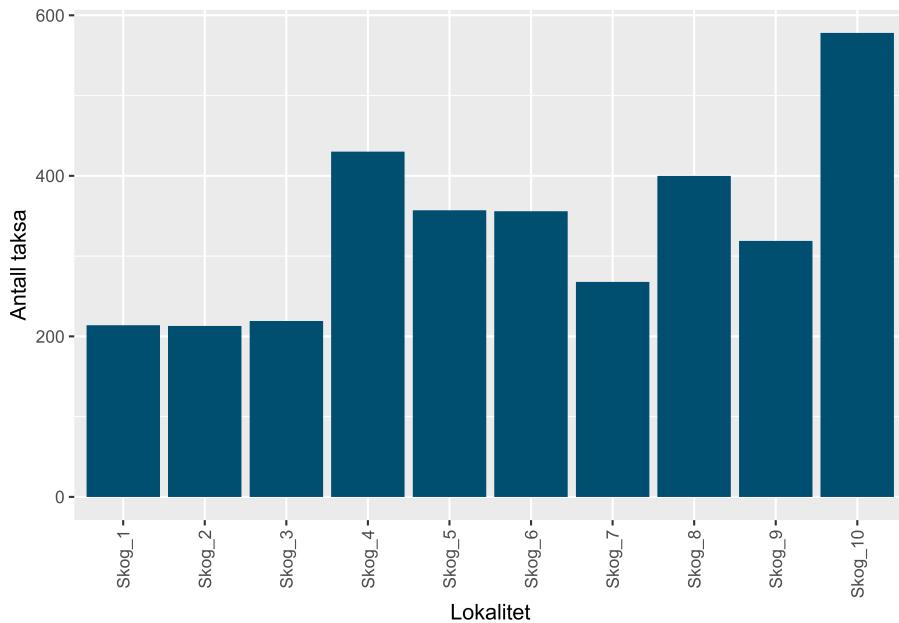


Figure 2: Cumulative number of taxa found in window traps.

```

group_by(locality, habitat_type) %>%
  select(genus,species) %>%
  distinct() %>%
  count() %>%
  ungroup() %>%
  ggplot(.) +
  geom_bar(aes(y = n, x = locality, fill = habitat_type),
           stat = "identity") +
#  ggttitle(paste0("Kumulativ artsrikedom i\nalle vindusfeller per ", max(dat$dateemptied),
#                 theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
#                       legend.position = "none") +
#                     scale_fill_nina() +
#                     ylab("Antall taksa") +
#                     xlab("Lokalitet"))

## Adding missing grouping variables: `locality`, `habitat_type`
```

## Basic biomass graphs

Quick look at the biomass at the different sites.

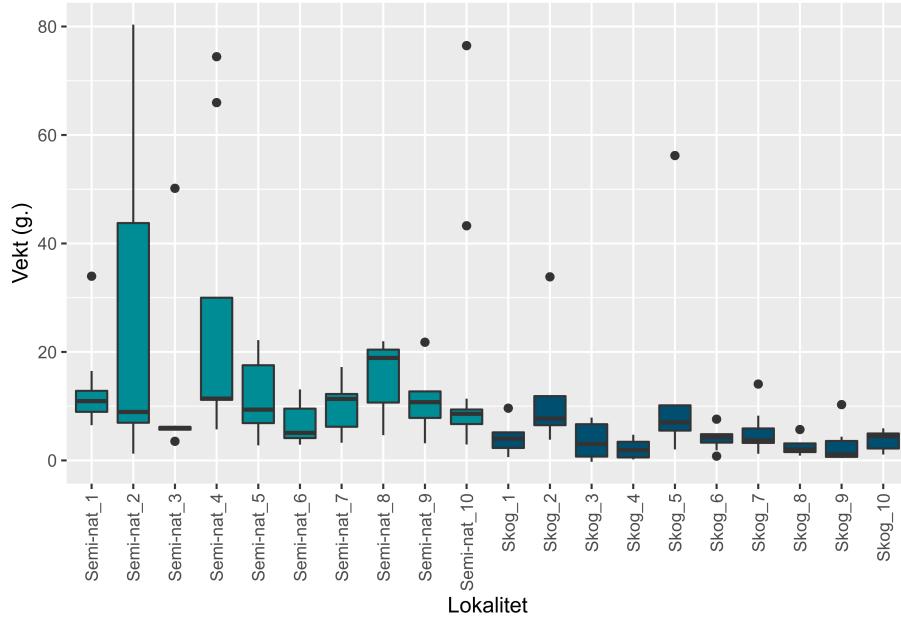


Figure 3: Average biomass (dry weight) in malaise traps.

```

dat %>%
  filter(grepl("MF", trap)) %>%
  group_by(locality, habitat_type) %>%
  ggplot(.) +
  geom_boxplot(aes(y = Toerrvekt_.g., x = locality, fill = habitat_type)) +
# ggtitle(paste0("Middels avrunnen vekt (pluss flaske)\nav insekter i malaisefeller per ",
# theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
#       legend.position = "none") +
#       scale_fill_nina() +
ylab("Vekt (g.)") +
xlab("Lokalitet")

## Warning: Removed 2834 rows containing non-finite values (stat_boxplot).

dat %>%
  filter(grepl("VF", trap)) %>%
  group_by(locality, habitat_type) %>%
  ggplot(.) +
  geom_boxplot(aes(y = Toerrvekt_.g., x = locality, fill = habitat_type), na.rm = T) +
# ggtitle(paste0("Middels avrunnen vekt (pluss flaske) av insekter i\nvindusfeller per ",
# theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
#       legend.position = "none") +

```

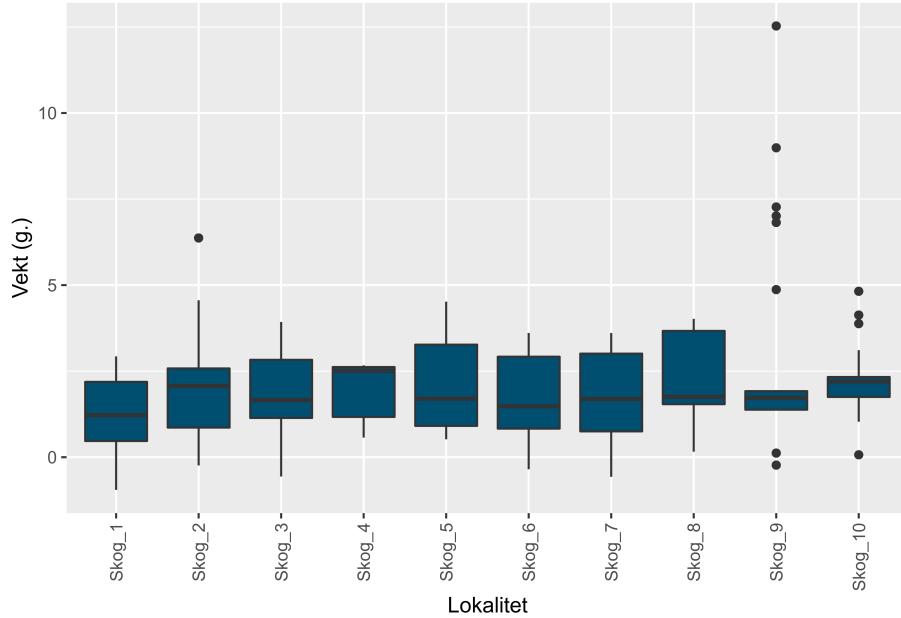


Figure 4: Average biomass (dry weight) in window traps.

```
scale_fill_nina() +
ylab("Vekt (g.)") +
xlab("Lokalitet")
```

Check out the high values. Nothing obviously suspicious.

```
dat %>%
  filter(grep("VF", trap)) %>%
  filter(Toerrvekt_.g. >5) %>%
  group_by(GenlabID) %>%
  summarise(mean_toerr_fl = mean(Vekt_flaske_.insekter_toerr_.g.),
            mean_tom_fl = mean(Vekt_tom_flaske_.g.),
            locality = first(locality),
            empty_week = mean(empty_week))

## `summarise()` ungrouping output (override with ` .groups` argument)

## # A tibble: 6 x 5
##   GenlabID      mean_toerr_fl mean_tom_fl locality empty_week
##   <chr>          <dbl>        <dbl> <fct>       <dbl>
## 1 OVERV_20_031    77.0         70.6 Skog_2        24
## 2 OVERV_20_044    77.9         70.6 Skog_9        22
## 3 OVERV_20_045    77.6         70.6 Skog_9        24
```

```

## 4 OVERV_20_046      77.4      70.6 Skog_9      24
## 5 OVERV_20_048      79.6      70.6 Skog_9      24
## 6 OVERV_20_049      83.1      70.6 Skog_9      24

```

## Check overlap between malaise and window traps

Total number of coleoptera species per trap type

```

col_spec_richn_tot <- dat %>%
  filter(order == "Coleoptera",
         habitat_type == "Skog") %>%
  group_by(locality) %>%
  summarize(col_spec_richn = n_distinct(genus, species)) %>%
  mutate(trap_type = "Alle")

## `summarise()` ungrouping output (override with `.`groups` argument)
col_spec_richn_trap <- dat %>%
  filter(order == "Coleoptera",
         habitat_type == "Skog") %>%
  group_by(locality,
           trap_type) %>%
  summarize(col_spec_richn = n_distinct(genus, species))

## `summarise()` regrouping output by 'locality' (override with `.`groups` argument)
col_spec_richn <- col_spec_richn_tot %>%
  union_all(col_spec_richn_trap)

ggplot(col_spec_richn, aes(y = col_spec_richn,
                           x = locality,
                           fill = trap_type)) +
  geom_bar(stat = "identity",
            position = position_dodge()) +
#  ggtitle(paste0("Kumulativt antall billearter (identifiserte taksa)\ni hver felletyp per"))
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  scale_fill_nina(name = "Felletype") +
  ylab("Antall taksa") +
  xlab("Lokalitet")

```

## Percentage of catches in each trap type per species

How to make it more readable? Split out all the singletons into a separate graph?

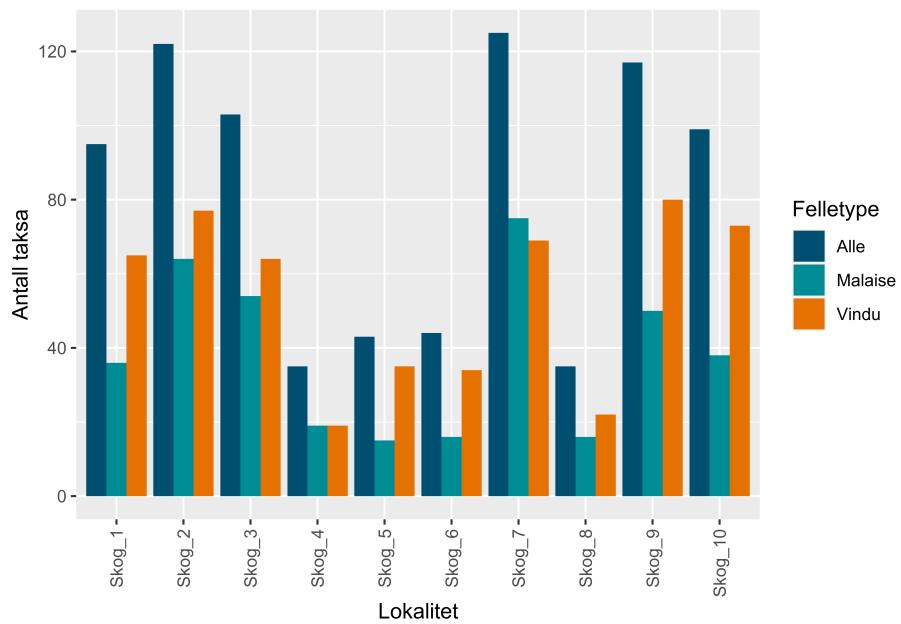


Figure 5: Cumulative number of Coleoptera taxa identified in each trap type

```

col_perc_traptype <- dat %>%
  filter(order == "Coleoptera",
         habitat_type == "Skog") %>%
  group_by(trap_type,
           genus,
           species_latin) %>%
  summarize(no_catches = n())

## `summarise()` regrouping output by 'trap_type', 'genus' (override with `groups` argument)
to_plot_temp <- col_perc_traptype %>%
  ungroup() %>%
  arrange(species_latin,
          trap_type) %>%
  filter(no_catches > 1) %>%
  group_by(species_latin) %>%
  mutate(caught_in = n_distinct(trap_type))

temp_1 <- to_plot_temp %>%
  filter(caught_in == 1) %>%
  arrange(trap_type, species_latin)

temp_2 <- to_plot_temp %>%
  filter(caught_in == 2) %>%
  arrange(species_latin, trap_type)

to_plot <- rbind(temp_1, temp_2) %>%
  ungroup() %>%
  mutate(trap_type = factor(trap_type),
         species_latin = factor(species_latin),
         facet = rep(c(1, 2, 3, 4), times = c(61, 60, 60, 60)))

to_plot %>%
ggplot(.) +
  geom_bar(aes(y = reorder(species_latin, species_latin), x = no_catches, fill = trap_type),
            stat = "identity",
            position = position_dodge()) +
  scale_fill_nina(name = "Trap type") +
  ylab("") +
  xlab("Number of catches") +
  theme(legend.position = "none",
        text = element_text(size=5)) +
  facet_wrap(facets = vars(facet),
             scales = "free_y",
             strip.position = "right",
             ncol = 4)

```



```
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA
```

```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA
```



```
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA
```

```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA
```



```
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA
```

```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA
```



```
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA
```

```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA
```



```
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA
```

```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA
```



```
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA
```

```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA
```



```
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA
```

```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA
```



```
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA
```

```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA
```



```
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA
```

```

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

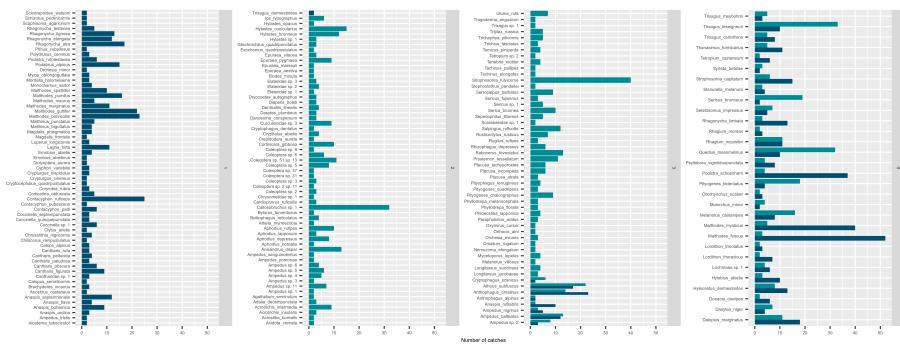
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

```



```

p1 <- to_plot %>%
  slice(1:50) %>%
  group_by(trap_type) %>%
  ggplot(.) +
  geom_bar(aes(y = species_latin, x = no_catches, fill = trap_type),
           stat = "identity",
           position = position_dodge()) +
  scale_fill_nina(name = "Trap type") +
  ylab("") +
  xlab("Number of catches") +
  theme(legend.position = "none",
        text = element_text(size=5))

```

```

p2 <- to_plot %>%
  slice(51:100) %>%
  group_by(trap_type) %>%
ggplot(.) +
  geom_bar(aes(y = species_latin, x = no_catches, fill = trap_type),
            stat = "identity",
            position = position_dodge()) +
  scale_fill_nina(name = "Trap type") +
  ylab("") +
  xlab("Number of catches") +
  theme(legend.position = "none",
        text = element_text(size=5))

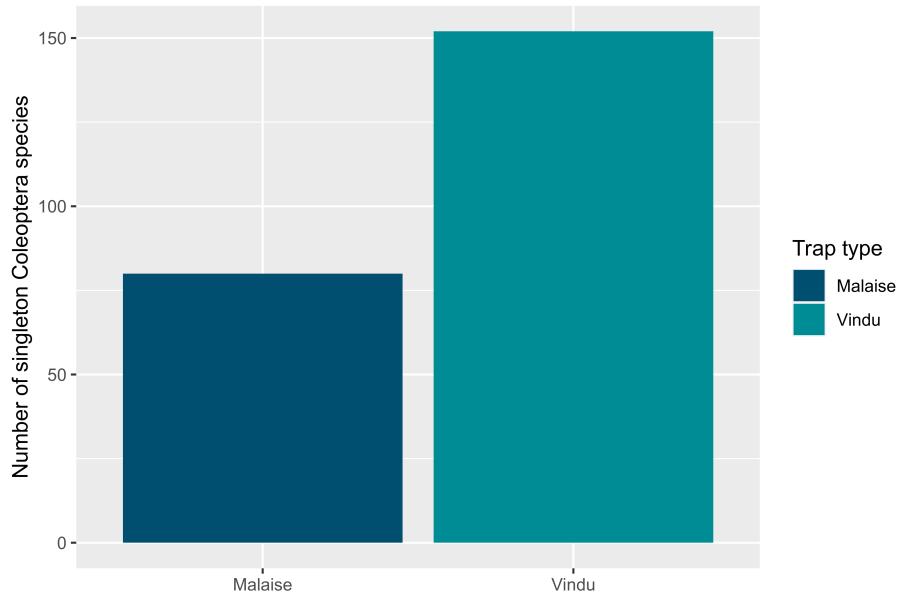
p3 <- to_plot %>%
  slice(101:150) %>%
  group_by(trap_type) %>%
ggplot(.) +
  geom_bar(aes(y = species_latin, x = no_catches, fill = trap_type),
            stat = "identity",
            position = position_dodge()) +
  scale_fill_nina(name = "Trap type") +
  ylab("") +
  xlab("Number of catches") +
  theme(legend.position = "none",
        text = element_text(size=5))

p4 <- to_plot %>%
  slice(151:200) %>%
  group_by(trap_type) %>%
ggplot(.) +
  geom_bar(aes(y = species_latin, x = no_catches, fill = trap_type),
            stat = "identity",
            position = position_dodge()) +
  scale_fill_nina(name = "Trap type") +
  ylab("") +
  xlab("Number of catches") +
  theme(text = element_text(size=5))

grid.arrange(p1, p2, p3, p4, nrow = 1)

```





## Figs to Niclas on Pollinators

Niclas Gyllenstrand at NRM asked how efficient the malaisetrap + metabarcoding are at finding pollinators. This could be a relevant graph for the report as well.

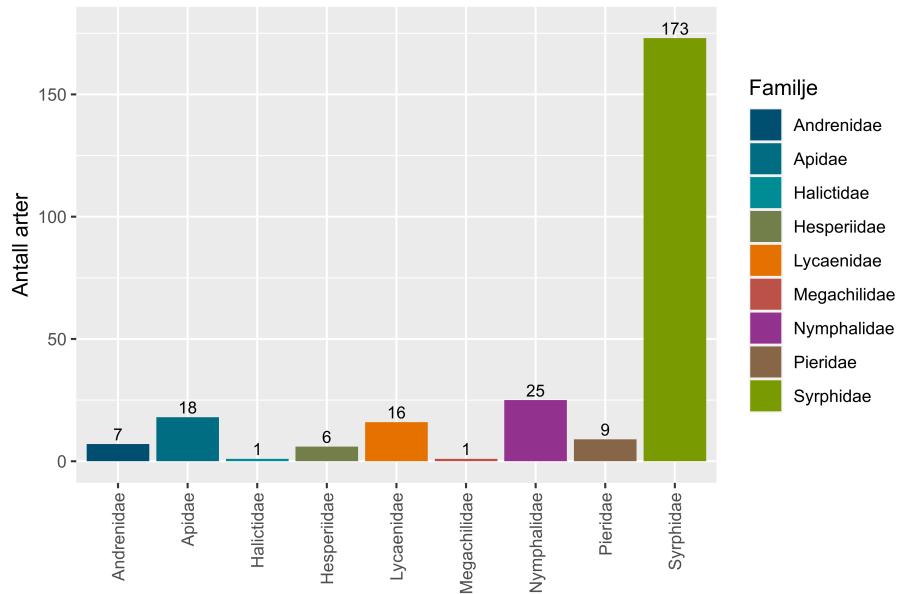
```
poll_spec_richn <- dat %>%
  filter(trap_type == "Malaise") %>%
  filter(family == "Andrenidae" |
         family == "Apidae" |
         family == "Colletidae" |
         family == "Halictidae" |
         family == "Megachilidae" |
         family == "Melittidae" |
         family == "Syrphidae" |
         family == "Hesperiidae" |
         family == "Lycaenidae" |
         family == "Nymphalidae" |
         family == "Papilionidae" |
         family == "Pieridae" |
         family == "Riodinidae")
  ) %>%
```

```

group_by(family) %>%
distinct(family, genus, species) %>%
summarize(col_spec_richn = n(),
          no_unidentified_sp = sum(grepl("sp.", species)))

## `summarise()` ungrouping output (override with `groups` argument)
ggplot(poll_spec_richn, aes(x = family, y = col_spec_richn, label = col_spec_richn)) +
  geom_bar(stat = "identity",
            aes(fill = family)) +
  geom_text(size = 3,
            position = position_nudge(y = 4)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  scale_fill_nina(name = "Familje") +
  xlab("") +
  ylab("Antall arter")

```



```

pollinator_species <- dat %>%
  filter(trap_type == "Malaise") %>%
  filter(family == "Andrenidae" |
            family == "Apidae" |
            family == "Colletidae" |
            family == "Halictidae" |
            family == "Megachilidae" |
            family == "Melittidae" |
            family == "Syrphidae")

```

```

family == "Syrphidae" |
family == "Hesperiidae" |
family == "Lycaenidae" |
family == "Nymphalidae" |
family == "Papilionidae" |
family == "Pieridae" |
family == "Riodinidae"
) %>%
distinct(order, family, genus, species) %>%
arrange(family, genus, species)

dat %>%
filter(trap_type == "Malaise") %>%
filter(family == "Apidae") %>%
distinct(genus, species) %>%
select(genus, species)

## # A tibble: 18 x 2
##   genus   species
##   <chr>  <chr>
## 1 Bombus lucorum
## 2 Bombus pratorum
## 3 Bombus jonellus
## 4 Bombus hypnorum
## 5 Apis mellifera
## 6 Bombus norvegicus
## 7 Bombus ashtoni
## 8 Bombus sporadicus
## 9 Bombus soroeensis
## 10 Bombus pascuorum
## 11 Bombus consobrinus
## 12 Bombus cryptarum
## 13 Bombus cingulatus
## 14 Bombus hortorum
## 15 Bombus monticola
## 16 Bombus terrestris
## 17 Bombus magnus
## 18 Bombus wurflenii
write_csv(pollinator_species,
          path = "out/pollinator_species.csv")

## Warning: The `path` argument of `write_csv()` is deprecated as of readr 1.4.0.
## Please use the `file` argument instead.
## This warning is displayed once every 8 hours.

```

```
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

## Phenology graphs

Some “timeseries” throughout the season.

```
seasonAvgNo <- dat %>%
  filter(weeks_sampled == 2) %>%
  group_by(habitat_type, date_emptied, locality) %>%
  summarise(no_species = n_distinct(genus, species),
            sum_biomass = sum(Vekt_flaske_..insekter_vaat_.g., na.rm = T) / no_species)

## `summarise()` regrouping output by 'habitat_type', 'date_emptied' (override with `groups`)

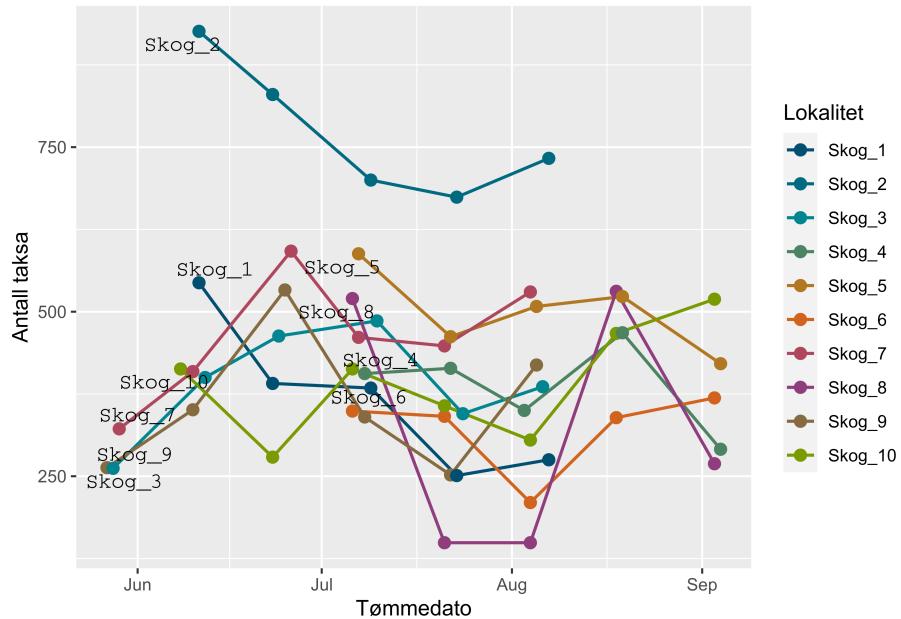
# test <- seasonAvgNo <- dat %>%
#   filter(weeks_sampled == 2) %>%
#   group_by(habitat_type, date_emptied, locality) %>%
#   select(habitat_type, date_emptied, locality, Vekt_flaske_..insekter_vaat_.g.) %>%
#   distinct() %>%
#   summarise(sum(Vekt_flaske_..insekter_vaat_.g., na.rm = T))

to_plot <- seasonAvgNo %>%
  filter(habitat_type == "Skog")

to_annotate <- to_plot %>%
  group_by(locality) %>%
  arrange(date_emptied) %>%
  filter(row_number()==1)

ggplot(to_plot) +
  geom_line(aes(x = date_emptied,
                y = no_species,
                color = locality),
            lwd = 0.75) +
  geom_point(aes(x = date_emptied,
                 y = no_species,
                 color = locality),
             size = 2.5) +
  geom_text_repel(data = to_annotate,
                  aes(x = date_emptied,
                      y = no_species,
                      label = locality,
                      family = "mono")) +
  ylab("Antall taksa") +
```

```
xlab("Tømmedato") +
scale_color_nina(name = "Lokalitet")
```



```
to_plot <- seasonAvgNo %>%
  filter(habitat_type == "SN_Gressmark")

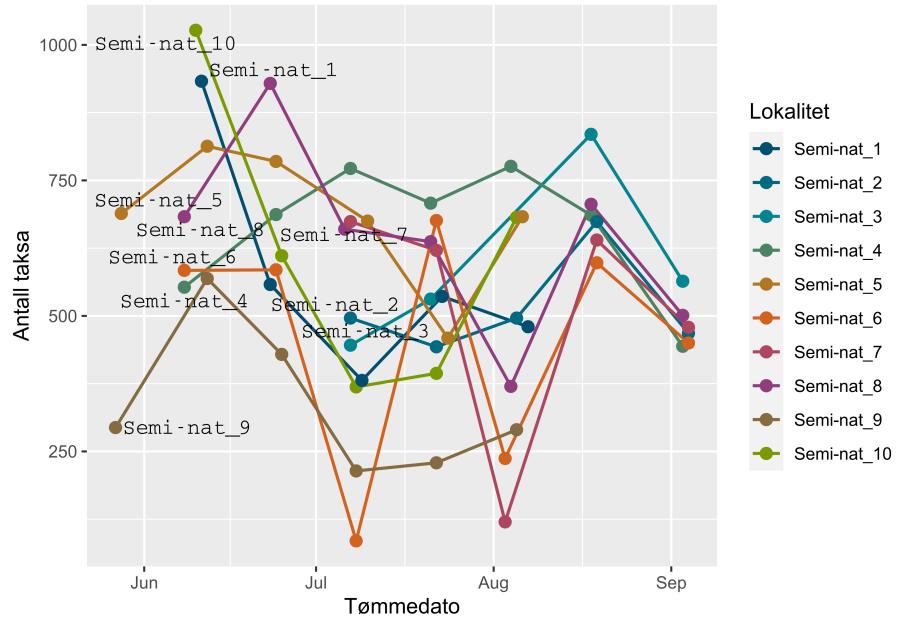
to_annotation <- to_plot %>%
  group_by(locality) %>%
  arrange(dateemptied) %>%
  filter(row_number() == 1)

ggplot(to_annotation) +
  geom_line(aes(x = dateemptied,
                y = no_species,
                color = locality),
            lwd = 0.75) +
  geom_point(aes(x = dateemptied,
                 y = no_species,
                 color = locality),
             size = 2.5) +
  geom_text_repel(data = to_annotation,
                  aes(x = dateemptied,
                      y = no_species,
                      label = locality,
                      family = "mono")) +
```

```

ylab("Antall taksa") +
xlab("Tømmedato") +
scale_color_nina(name = "Lokalitet")

```



```

to_plot <- seasonAvgNo %>%
  filter(habitat_type == "Skog")

to_annotation <- to_plot %>%
  group_by(locality) %>%
  arrange(dateemptied) %>%
  filter(row_number() == 1)

ggplot(to_plot) +
  geom_line(aes(x = dateemptied,
                y = sum_biomass,
                color = locality),
            lwd = 0.75) +
  geom_point(aes(x = dateemptied,
                 y = sum_biomass,
                 color = locality),
             size = 2.5) +
  geom_text_repel(data = to_annotation,
                  aes(x = dateemptied,
                      y = sum_biomass,
                      label = locality,

```

```

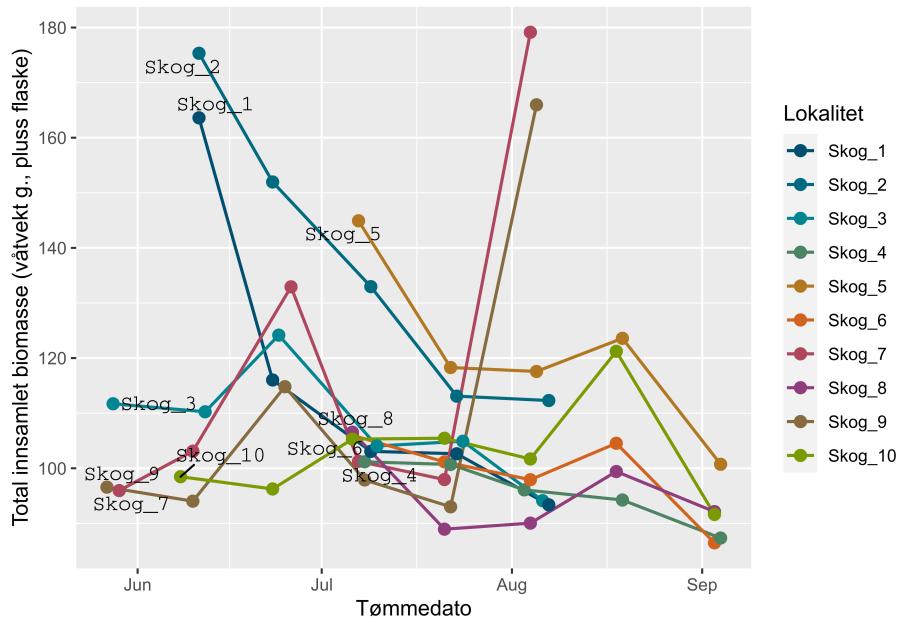
family = "mono") +  

ylab("Total innsamlet biomasse (våtvekt g., pluss flaske)") +  

xlab("Tømmedato") +  

scale_color_nina(name = "Lokalitet")

```



```

to_plot <- seasonAvgNo %>%
  filter(habitat_type == "SN_Gressmark")

to_annotation <- to_plot %>%
  group_by(locality) %>%
  arrange(dateemptied) %>%
  filter(row_number() == 1)

ggplot(to_plot) +
  geom_line(aes(x = dateemptied,
                y = sum_biomass,
                color = locality),
            lwd = 0.75) +
  geom_point(aes(x = dateemptied,
                 y = sum_biomass,
                 color = locality),
             size = 2.5) +
  geom_text_repel(data = to_annotation,
                  aes(x = dateemptied,
                      y = sum_biomass,

```



## Plot some weather data

```

logger_tbl <- tbl(con,
                   in_schema("loggers", "logger_data")) %>%
  select(-id)

logger_dep_tbl <- tbl(con,
                      in_schema("loggers", "logger_deployments")) %>%
  select(-id)

logger_data <- logger_tbl %>%
  left_join(logger_dep_tbl,
            by = c("logger_id" = "logger_id",
                  "logger_type" = "logger_type")) %>%
  mutate(day = as.Date(date))

```

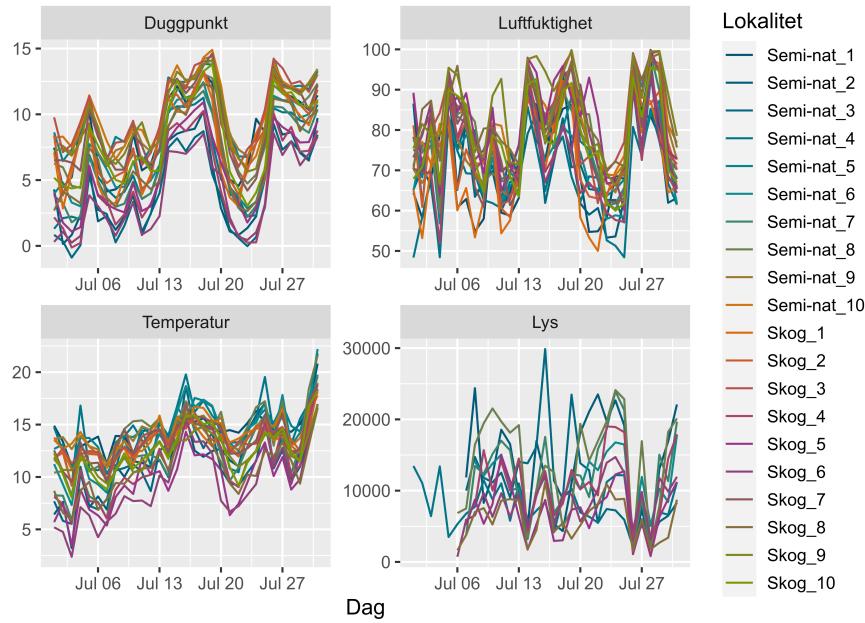
Plot average values for each site during July

```

tt <- logger_data %>%
  group_by(location, day, data_type) %>%
  summarize(value = mean(value)) %>%
  filter(day >= '2020-07-01' &
         day < '2020-08-01'
       ) %>%
  collect() %>%
  mutate(Lokalitet = factor(location, levels = c(paste0("Semi-nat_", 1:10), paste0("Skog_", 1:10)),
    mutate(data_type = factor(data_type, levels = ))
  ##Why can't I set the levels in the pipe?
  levels(tt$data_type) <- c("Duggpunkt", "Luftfuktighet", "Temperatur", "Lys")

tt %>%
  ggplot(.) +
  geom_line(aes(x = day, y = value, color = Lokalitet)) +
  facet_wrap(~data_type,
             scales = "free") +
  scale_color_nina(name = "Lokalitet") +
  xlab("Dag") +
  ylab("") +
  theme(legend.key.height = unit(0.5, "cm"))

```



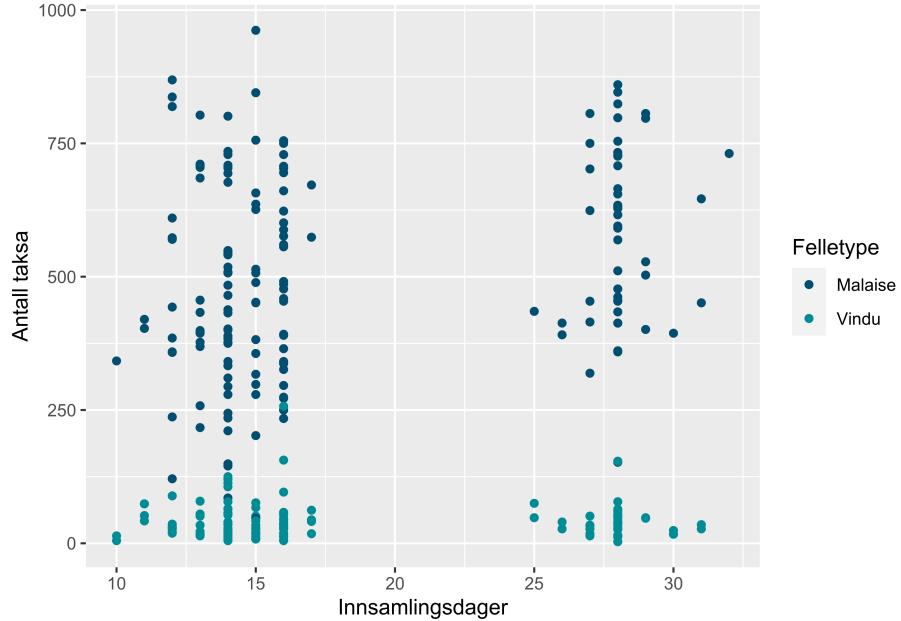
## Sampling time analysis

Time to look at the impact of sampling time, 2 vs 4 weeks.

First off, simple plot of species richness against days sampled.

```
dat %>%
  group_by(locality,
    trap,
    trap_type,
    empty_week,
    weeks_sampled,
    days_collected) %>%
  summarise(no_spec = n_distinct(order, family, genus, species)) %>%
  ggplot(aes(x = days_collected, y = no_spec)) +
  geom_point(aes(color = trap_type)) +
  scale_color_nina(name = "Felletype") +
  ylab("Antall taksa") +
  xlab("Innsamlingsdager")
```

## `summarise()` regrouping output by 'locality', 'trap', 'trap\_type', 'empty\_week', 'weeks\_sampled' into 11 groups



Not easy to see any patterns here, since there is such variation in the catches throughout the season. We need to sum up the catches in the 2 2-week periods and compare to the single 4 week periods.

We first identify which of the 2-week samples that should be aggregated.

```

#4-week "target" to agg to
week_4 <- dat %>%
  filter(weeks_sampled == 4) %>%
  select(locality,
         empty_week) %>%
  distinct() %>%
  mutate(agg_id = 1:n())

#add 2 weeks earlier
earlier_2_weeks <- week_4 %>%
  select(locality,
         agg_id) %>%
  mutate(empty_week = week_4$empty_week - 2)

which_to_agg <- week_4 %>%
  bind_rows(earlier_2_weeks)

#add aggregation ids to 2-week samplings
to_agg <- dat %>%
  filter(weeks_sampled == 2) %>%
  left_join(which_to_agg,
            by = c("locality" = "locality",
                  "empty_week" = "empty_week"))

```

Check which didn't get an aggregation

```

to_agg %>%
  filter(is.na(agg_id)) %>%
  select(locality,
         empty_week,
         trap) %>%
  distinct() %>%
  arrange(empty_week,
         locality) %>%
  print(n = Inf)

## # A tibble: 38 x 3
##   locality    empty_week trap
##   <fct>        <dbl> <chr>
## 1 Semi-nat_1     24  MF1
## 2 Semi-nat_1     24  MF2
## 3 Semi-nat_10    24  MF1
## 4 Semi-nat_10    24  MF2
## 5 Skog_1          24  MF1
## 6 Skog_1          24  MF2
## 7 Skog_1          24  VF1

```

```

## 8 Skog_1          24 VF2
## 9 Skog_1          24 VF3
## 10 Skog_1         24 VF4
## 11 Skog_2          24 MF1
## 12 Skog_2          24 MF2
## 13 Skog_2          24 VF1
## 14 Skog_2          24 VF2
## 15 Skog_2          24 VF3
## 16 Skog_2          24 VF4
## 17 Semi-nat_2       32 MF1
## 18 Semi-nat_2       34 MF1
## 19 Semi-nat_2       34 MF2
## 20 Semi-nat_2       36 MF1
## 21 Semi-nat_3       36 MF1
## 22 Semi-nat_4       36 MF1
## 23 Semi-nat_6       36 MF1
## 24 Semi-nat_7       36 MF1
## 25 Semi-nat_8       36 MF1
## 26 Skog_4           36 MF1
## 27 Skog_4           36 VF1
## 28 Skog_4           36 VF3
## 29 Skog_5           36 MF1
## 30 Skog_5           36 VF1
## 31 Skog_5           36 VF3
## 32 Skog_6           36 MF1
## 33 Skog_6           36 VF1
## 34 Skog_6           36 VF3
## 35 Skog_8           36 MF1
## 36 Skog_10          36 MF1
## 37 Skog_10          36 VF1
## 38 Skog_10          36 VF3

```

The samples from week 24 are expected, since not all traps where set up early enough to get 2 emptyings by then.

```

dat %>%
  filter(locality == 'Semi-nat_2' &
         empty_week == 34) %>%
  select(locality,
         empty_week,
         dateemptied,
         weeks_sampled,
         trap,
         GenlabID) %>%
  distinct()

## # A tibble: 3 x 6

```

```

##   locality   empty_week date_emptied weeks_sampled trap  GenlabID
##   <fct>          <dbl> <date>           <dbl> <chr> <chr>
## 1 Semi-nat_2      34 2020-08-19        2 MF1  OVERRV_20_236_1
## 2 Semi-nat_2      34 2020-08-19        2 MF1  OVERRV_20_236_2
## 3 Semi-nat_2      34 2020-08-19        2 MF2  OVERRV_20_237

But Semi-nat_1, week_32, trap MF2 should be there. Semi-nat_5 as well.
Seems that these haven't been run yet. I'll remove these for the comparison.

#Sum the genetic reads to combine the two weeks
agg_data_2_week <- to_agg %>%
  filter(!is.na(agg_id)) %>%
  group_by(locality,
           trap_type,
           trap,
           agg_id,
           species_latin) %>%
  summarise(value = sum(value)) %>%
  ungroup()

## `summarise()` regrouping output by 'locality', 'trap_type', 'trap', 'agg_id' (override w...
#Add the week information for the second sampling
agg_data_2_week <- agg_data_2_week %>%
  left_join(week_4,
            by = c("agg_id" = "agg_id",
                  "locality" = "locality")) %>%
  mutate(weeks_sampled = 2) %>%
  select(locality,
         empty_week,
         weeks_sampled,
         trap_type,
         trap,
         species_latin,
         value)

#Select only the original 4 week samplings
data_4_week <- dat %>%
  filter(weeks_sampled == 4) %>%
  select(locality,
         empty_week,
         weeks_sampled,
         trap_type,
         trap,
         species_latin,
         value)

```

```
#Add the original 4 weeks to the summarized 2 2-week samplings
week_comp_data <- agg_data_2_week %>%
  bind_rows(data_4_week)
```

Summarise the number of species in both sample lengths.

```
no_species_weeks <- week_comp_data %>%
  group_by(locality,
           empty_week,
           weeks_sampled,
           trap_type) %>%
  summarise(no_species = n())
```

```
## `summarise()` regrouping output by 'locality', 'empty_week', 'weeks_sampled' (override w
```

```
#Check comparison
```

```
no_species_weeks %>%
  group_by(locality,
           empty_week,
           trap_type) %>%
  summarise(no_comp = n()) %>%
  print(n = Inf)
```

```
## `summarise()` regrouping output by 'locality', 'empty_week' (override with `groups` argu
```

```
## # A tibble: 72 x 4
##   locality   empty_week trap_type no_comp
##   <fct>       <dbl> <chr>     <int>
## 1 Semi-nat_1    28 Malaise      2
## 2 Semi-nat_1    32 Malaise      2
## 3 Semi-nat_2    30 Malaise      2
## 4 Semi-nat_3    30 Malaise      2
## 5 Semi-nat_3    34 Malaise      2
## 6 Semi-nat_4    26 Malaise      2
## 7 Semi-nat_4    30 Malaise      2
## 8 Semi-nat_4    34 Malaise      2
## 9 Semi-nat_5    24 Malaise      2
## 10 Semi-nat_5   28 Malaise      2
## 11 Semi-nat_5   32 Malaise      2
## 12 Semi-nat_6   26 Malaise      2
## 13 Semi-nat_6   30 Malaise      2
## 14 Semi-nat_6   34 Malaise      2
## 15 Semi-nat_7   30 Malaise      2
## 16 Semi-nat_7   34 Malaise      2
## 17 Semi-nat_8   26 Malaise      2
## 18 Semi-nat_8   30 Malaise      2
```

## 19	Semi-nat_8	34	Malaise	2
## 20	Semi-nat_9	24	Malaise	2
## 21	Semi-nat_9	28	Malaise	2
## 22	Semi-nat_9	32	Malaise	2
## 23	Semi-nat_10	28	Malaise	2
## 24	Semi-nat_10	32	Malaise	2
## 25	Skog_1	28	Malaise	2
## 26	Skog_1	28	Vindu	2
## 27	Skog_1	32	Malaise	2
## 28	Skog_1	32	Vindu	1
## 29	Skog_2	28	Malaise	2
## 30	Skog_2	28	Vindu	2
## 31	Skog_2	32	Malaise	2
## 32	Skog_2	32	Vindu	1
## 33	Skog_3	24	Malaise	2
## 34	Skog_3	24	Vindu	2
## 35	Skog_3	28	Malaise	2
## 36	Skog_3	28	Vindu	2
## 37	Skog_3	32	Malaise	2
## 38	Skog_3	32	Vindu	1
## 39	Skog_4	30	Malaise	2
## 40	Skog_4	30	Vindu	2
## 41	Skog_4	34	Malaise	2
## 42	Skog_4	34	Vindu	2
## 43	Skog_5	30	Malaise	2
## 44	Skog_5	30	Vindu	2
## 45	Skog_5	34	Malaise	2
## 46	Skog_5	34	Vindu	2
## 47	Skog_6	30	Malaise	2
## 48	Skog_6	30	Vindu	2
## 49	Skog_6	34	Malaise	2
## 50	Skog_6	34	Vindu	2
## 51	Skog_7	24	Malaise	2
## 52	Skog_7	24	Vindu	2
## 53	Skog_7	28	Malaise	2
## 54	Skog_7	28	Vindu	2
## 55	Skog_7	32	Malaise	2
## 56	Skog_7	32	Vindu	1
## 57	Skog_8	30	Malaise	2
## 58	Skog_8	30	Vindu	2
## 59	Skog_8	34	Malaise	2
## 60	Skog_8	34	Vindu	2
## 61	Skog_9	24	Malaise	2
## 62	Skog_9	24	Vindu	2
## 63	Skog_9	28	Malaise	2
## 64	Skog_9	28	Vindu	2

```

## 65 Skog_9           32 Malaise      2
## 66 Skog_9           32 Vindu        1
## 67 Skog_10          26 Malaise      2
## 68 Skog_10          26 Vindu        2
## 69 Skog_10          30 Malaise      2
## 70 Skog_10          30 Vindu        2
## 71 Skog_10          34 Malaise      2
## 72 Skog_10          34 Vindu        2

```

Make a comparison table.

```

comp_species_week <- no_species_weeks %>%
  pivot_wider(id_cols = c(locality,
                           empty_week,
                           trap_type),
              names_from = weeks_sampled,
              values_from = no_species,
              names_prefix = "sample_time_") %>%
  mutate(diff_sample_time = sample_time_4 - sample_time_2,
        prc_diff_sample_time = 100 * (sample_time_4 - sample_time_2) / sample_time_2)

```

Quality check of a few rows. Seems OK.

```

dat %>%
  filter(locality == "Semi-nat_1" &
         (empty_week == 26 | empty_week == 28) &
         trap == "MF1"
    ) %>%
  summarize(no_spec = n_distinct(species_latin))

## # A tibble: 1 x 1
##   no_spec
##   <int>
## 1     770

dat %>%
  filter(locality == "Semi-nat_5" &
         (empty_week == 22 | empty_week == 24) &
         trap == "MF1"
    ) %>%
  summarize(no_spec = n_distinct(species_latin))

## # A tibble: 1 x 1
##   no_spec
##   <int>
## 1    1216

dat %>%
  filter(locality == "Semi-nat_5" &

```

```

        (empty_week == 24 ) &
        trap == "MF2"
    ) %>%
summarize(no_spec = n_distinct(species_latin))

## # A tibble: 1 x 1
##   no_spec
##   <int>
## 1     646

dat %>%
  filter(locality == "Semi-nat_6" &
         (empty_week == 26 ) &
         trap == "MF2"
    ) %>%
summarize(no_spec = n_distinct(species_latin))

## # A tibble: 1 x 1
##   no_spec
##   <int>
## 1     824

week_comp_data %>%
  filter(locality == "Semi-nat_5" &
         (empty_week == 24 ) &
         trap == "MF2"
    ) %>%
summarize(no_spec = n_distinct(species_latin))

## # A tibble: 1 x 1
##   no_spec
##   <int>
## 1     646

no_species_weeks %>%
  filter(locality == "Semi-nat_5" &
         empty_week == 24)

## # A tibble: 2 x 5
## Groups:   locality, empty_week, weeks_sampled [2]
##   locality   empty_week weeks_sampled trap_type no_species
##   <fct>          <dbl>       <dbl> <chr>      <int>
## 1 Semi-nat_5      24           2 Malaise      1216
## 2 Semi-nat_5      24           4 Malaise      646

comp_species_week %>%
  group_by(locality) %>%
  filter(trap_type == "Malaise") %>%

```

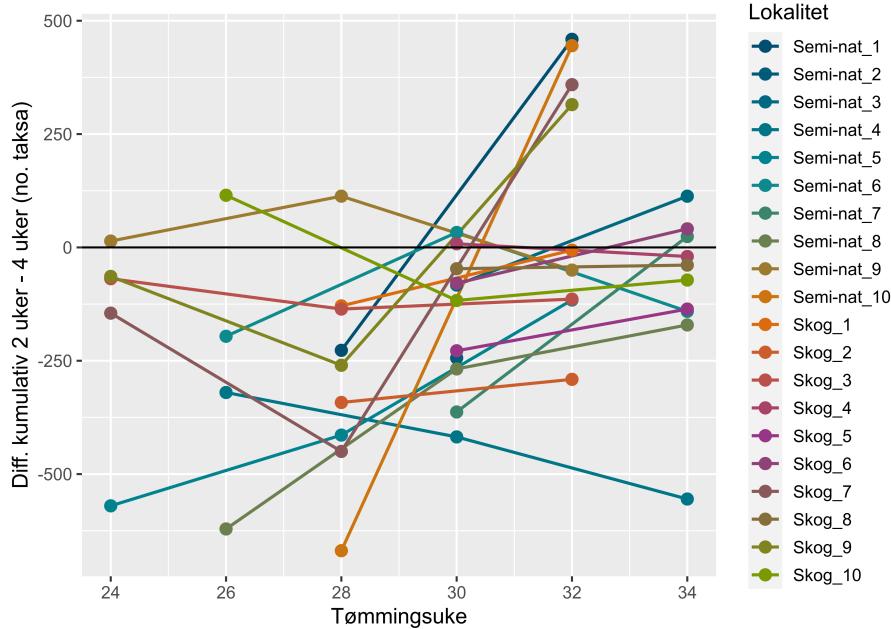


Figure 6: Difference (g.) in cumulative species richness over 2 2-week samplings, compared to 1 4 week sampling. Catch in 1 Malaise trap per period. Negative values indicate that 4-week samplings gave lower insect species richness than the two 2-week samplings.

```

ggplot(.) +
  geom_line(aes(x = empty_week,
                 y = diff_sample_time,
                 color = locality),
            lwd = 0.75) +
  geom_point(aes(x = empty_week,
                 y = diff_sample_time,
                 color = locality),
             size = 2.5) +
  scale_color_nina(name = "Lokalitet") +
  ylab("Diff. kumulativ 2 uker - 4 uker (no. taksa)") +
  xlab("Tømmingsuke") +
  geom_hline(yintercept = 0) +
  theme(legend.key.height = unit(0.5, "cm"))

comp_species_week %>%
  group_by(locality) %>%
  filter(trap_type == "Malaise") %>%
  ggplot(.) +

```

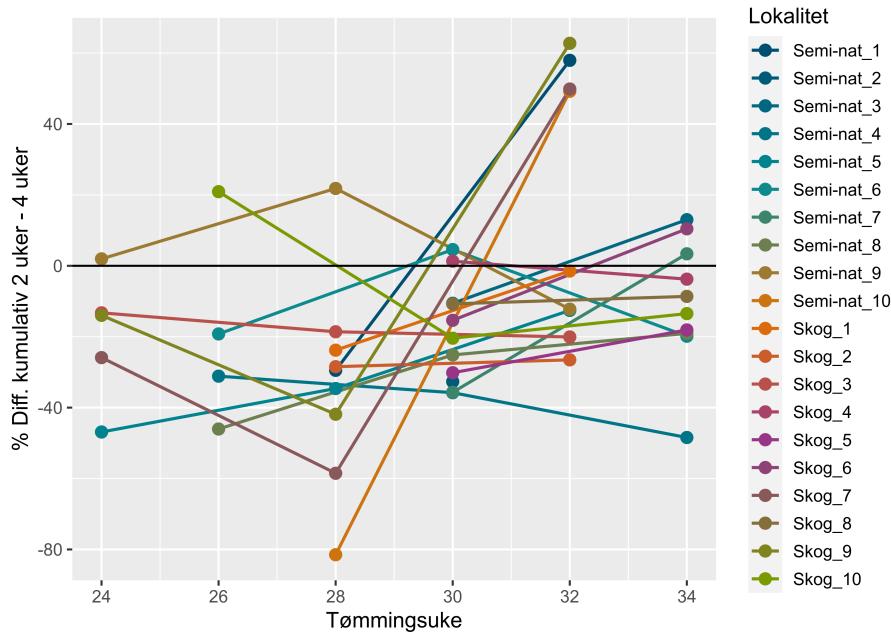


Figure 7: Percentage difference in cumulative species richness over 2 2-week samplings, compared to 1 4 week sampling. Catch in 1 Malaise trap per period. Negative values indicate that 4-week samplings gave lower insect species richness than the two 2-week samplings.

```

geom_line(aes(x = empty_week,
              y = prc_diff_sample_time,
              color = locality),
          lwd = 0.75) +
geom_point(aes(x = empty_week,
               y = prc_diff_sample_time,
               color = locality),
            size = 2.5) +
scale_color_nina(name = "Lokalitet") +
ylab("% Diff. kumulativ 2 uker - 4 uker") +
xlab("Tømmingsuke") +
geom_hline(yintercept = 0) +
theme(legend.key.height = unit(0.5, "cm"))

comp_species_week %>%
  group_by(locality) %>%
  filter(trap_type == "Vindu") %>%
  ggplot(.) +
  geom_line(aes(x = empty_week,

```

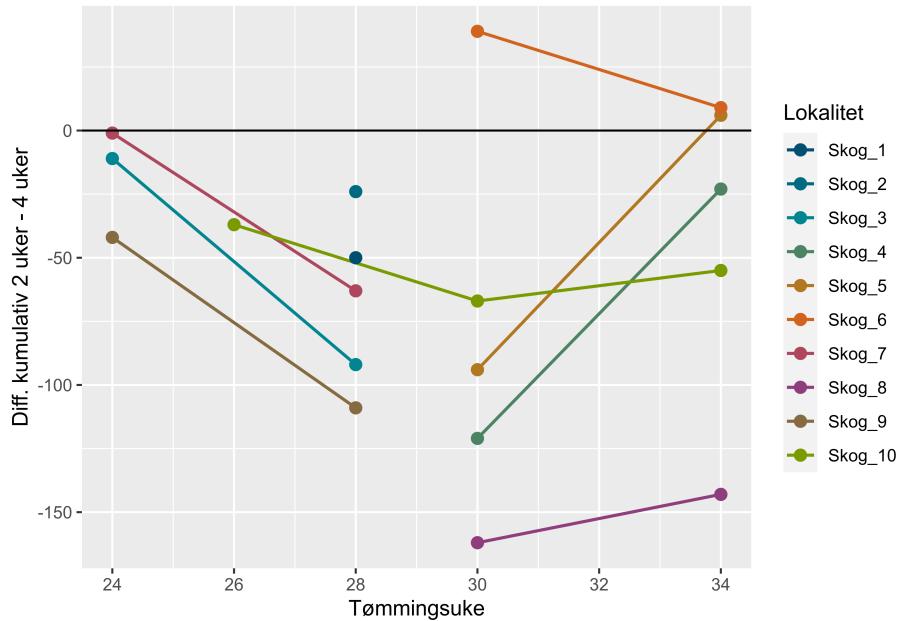


Figure 8: Difference in cumulative species richness over 2 2-week samplings, compared to 1 4 week sampling. Catch in 1 Malaise trap per period. Negative values indicate that 4-week samplings gave lower insect species richness than the two 2-week samplings.

```

y = diff_sample_time,
color = locality),
lwd = 0.75) +
geom_point(aes(x = empty_week,
y = diff_sample_time,
color = locality),
size = 2.5) +
scale_color_nina(name = "Lokalitet") +
ylab("Diff. kumulativ 2 uker - 4 uker") +
xlab("Tømmingsuke") +
geom_hline(yintercept = 0)

## Warning: Removed 5 row(s) containing missing values (geom_path).
## Warning: Removed 5 rows containing missing values (geom_point).

comp_species_week %>%
  group_by(locality) %>%
  filter(trap_type == "Vindu") %>%
  ggplot(.) +

```

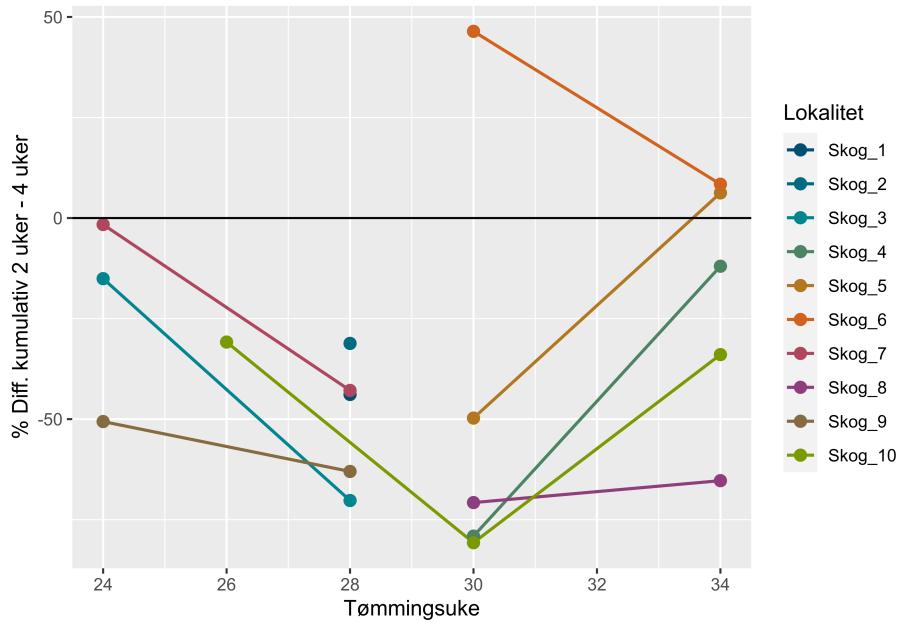


Figure 9: Percentage difference in cumulative species richness over 2 2-week samplings, compared to 1 4 week sampling. Catch in 2 window traps per period. Negative values indicate that 4-week samplings gave lower insect species richness than the two 2-week samplings.

```

geom_line(aes(x = empty_week,
              y = prc_diff_sample_time,
              color = locality),
          lwd = 0.75) +
  geom_point(aes(x = empty_week,
                 y = prc_diff_sample_time,
                 color = locality),
             size = 2.5) +
  scale_color_nina(name = "Lokalitet") +
  ylab("% Diff. kumulativ 2 uker - 4 uker") +
  xlab("Tømmingsuke") +
  geom_hline(yintercept = 0)

## Warning: Removed 5 row(s) containing missing values (geom_path).
## Warning: Removed 5 rows containing missing values (geom_point).

```

A similar comparison, but for biomass.

```
biomass_agg_data_2_week <- to_agg %>%
  filter(!is.na(agg_id)) %>%
  select(locality,
         empty_week,
         trap_type,
         trap,
         agg_id,
         biomass = Toerrvekt_.g.) %>%
  distinct() %>%
  arrange(locality,
          empty_week,
          trap) %>%
  group_by(locality,
           trap_type,
           agg_id) %>%
  summarise(empty_week = max(empty_week),
            biomass = sum(biomass)
            ) %>%
  ungroup() %>%
  select(-agg_id)

## `summarise()` regrouping output by 'locality', 'trap_type' (override with `.`groups` argument)
biomass_data_4_week <- dat %>%
  filter(weeks_sampled == 4) %>%
  select(locality,
         empty_week,
         trap_type,
         trap,
         biomass = Toerrvekt_.g.) %>%
  distinct() %>%
  arrange(locality,
          empty_week,
          trap) %>%
  group_by(locality,
           empty_week,
           trap_type) %>%
  summarise(empty_week = max(empty_week),
            biomass = sum(biomass)
            ) %>%
  ungroup()

## `summarise()` regrouping output by 'locality', 'empty_week' (override with `.`groups` argument)
```

```

#Make comparison table
biomass_comp <- biomass_agg_data_2_week %>%
  left_join(biomass_data_4_week,
            by = c("locality" = "locality",
                   "empty_week" = "empty_week",
                   "trap_type" = "trap_type"),
            suffix = c("_2_weeks", "_4_weeks")) %>%
  mutate(diff_sample_time = biomass_4_weeks - biomass_2_weeks,
         prc_diff_sample_time = 100 * (biomass_4_weeks - biomass_2_weeks) / biomass_2_weeks)

biomass_comp %>%
  group_by(locality) %>%
  filter(trap_type == "Malaise") %>%
  ggplot(.) +
  geom_line(aes(x = empty_week,
                 y = prc_diff_sample_time,
                 color = locality),
            lwd = 0.75) +
  geom_point(aes(x = empty_week,
                 y = prc_diff_sample_time,
                 color = locality),
             size = 2.5) +
  scale_color_nina(name = "Lokalitet") +
  ylab("% Diff. kumulativ 2 uker - 4 uker") +
  xlab("Tømmingsuke") +
  geom_hline(yintercept = 0) +
  theme(legend.key.height = unit(0.5, "cm"))

## Warning: Removed 3 row(s) containing missing values (geom_path).
## Warning: Removed 3 rows containing missing values (geom_point).

We can do the same comparison with wet weight, although we might expect
that this will add some weight of the ethanol and the bottles, which will lead
to bias when we sum these together for the 2 weeks. Anyway, this is the way it
looks for wet weight.

biomass_agg_data_2_week <- to_agg %>%
  filter(!is.na(agg_id)) %>%
  select(locality,
         empty_week,
         trap_type,
         trap,
         agg_id,
         biomass = Vaatvekt_.g.) %>%
  distinct() %>%
  arrange(locality,

```

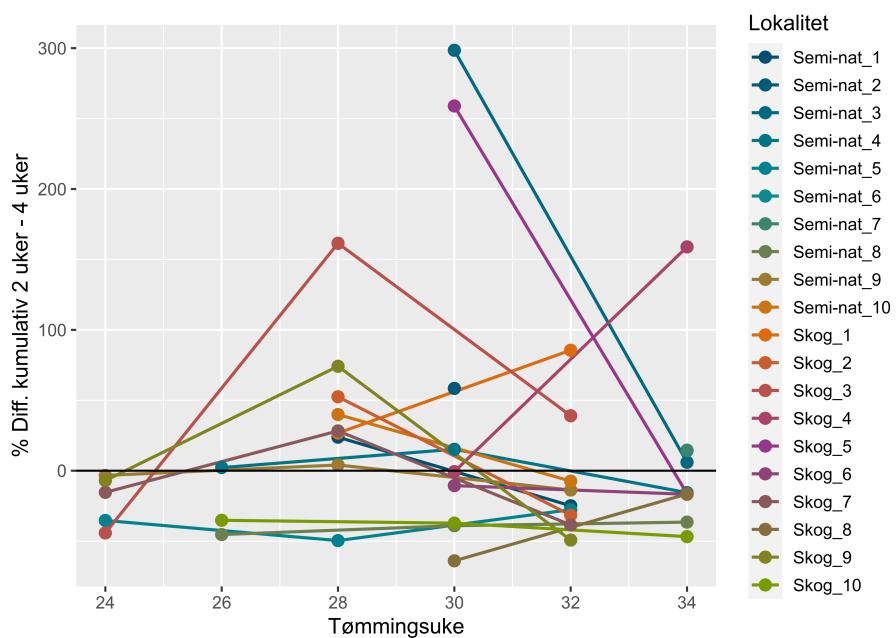


Figure 10: Percentage difference in cumulative dry weight over 2 2-week samplings, compared to 1 4 week sampling. Catch in 1 Malaise trap per period. Negative values indicate that 4-week samplings gave lower insect species richness than the two 2-week samplings.

```

        empty_week,
        trap) %>%
group_by(locality,
         trap_type,
         agg_id) %>%
summarise(empty_week = max(empty_week),
           biomass = sum(biomass)
         ) %>%
ungroup() %>%
select(-agg_id)

## `summarise()` regrouping output by 'locality', 'trap_type' (override with `groups` argument)
biomass_data_4_week <- dat %>%
  filter(weeks_sampled == 4) %>%
  select(locality,
         empty_week,
         trap_type,
         trap,
         biomass = Vaatvekt_.g.) %>%
distinct() %>%
arrange(locality,
        empty_week,
        trap) %>%
group_by(locality,
         empty_week,
         trap_type) %>%
summarise(empty_week = max(empty_week),
           biomass = sum(biomass)
         ) %>%
ungroup()

## `summarise()` regrouping output by 'locality', 'empty_week' (override with `groups` argument)
#Make comparison table
biomass_comp <- biomass_agg_data_2_week %>%
  left_join(biomass_data_4_week,
            by = c("locality" = "locality",
                  "empty_week" = "empty_week",
                  "trap_type" = "trap_type"),
            suffix = c("_2_weeks", "_4_weeks")) %>%
  mutate(diff_sample_time = biomass_2_weeks - biomass_4_weeks,
         prc_diff_sample_time = (biomass_2_weeks - biomass_4_weeks) / biomass_2_weeks)

biomass_comp %>%
  group_by(locality) %>%
  filter(trap_type == "Malaise") %>%

```

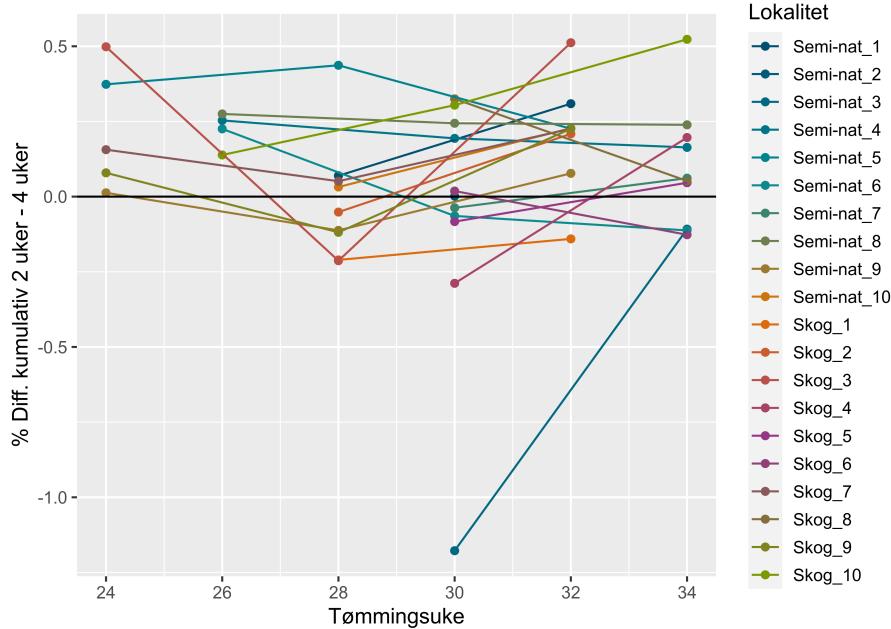


Figure 11: Percentage difference in cumulative wet weight over 2 2-week samplings, compared to 1 4 week sampling. Catch in 1 Malaise trap per period. Negative values indicate that 4-week samplings gave lower insect species richness than the two 2-week samplings.

```
ggplot(.) +
  geom_point(aes(x = empty_week, y = prc_diff_sample_time, color = locality)) +
  geom_line(aes(x = empty_week, y = prc_diff_sample_time, color = locality)) +
  scale_color_nina(name = "Lokalitet") +
  ylab("% Diff. kumulativ 2 uker - 4 uker") +
  xlab("Tømmingsuke") +
  geom_hline(yintercept = 0) +
  theme(legend.key.height = unit(0.5, "cm"))
```

## Compare the ethanol to the propglykol window traps

Quick analysis.

```
liquid_comp <- dat %>%
  filter(trap_type == "Vindu") %>%
  group_by(locality,
```

```

    trap,
    liquid,
    weeks_sampled,
    empty_week,
    sample_id
) %>%
summarise(no_species = n_distinct(species_latin))

## `summarise()` regrouping output by 'locality', 'trap', 'liquid', 'weeks_sampled', 'empty_...
liquid_comp

## # A tibble: 129 x 7
## # Groups:   locality, trap, liquid, weeks_sampled, empty_week [129]
##   locality trap   liquid   weeks_sampled empty_week sample_id no_species
##   <fct>    <chr>  <chr>      <dbl>        <dbl> <chr>       <int>
## 1 Skog_1    VF1   Ethanol      2           24 skog_1_VF1_we~     37
## 2 Skog_1    VF1   Ethanol      2           26 skog_1_VF1_we~     30
## 3 Skog_1    VF1   Ethanol      2           28 skog_1_VF1_we~     27
## 4 Skog_1    VF1   Ethanol      2           30 skog_1_VF1_we~     24
## 5 Skog_1    VF2   Ethanol      2           24 skog_1_VF2_we~     58
## 6 Skog_1    VF2   Ethanol      4           28 skog_1_VF2_we~     26
## 7 Skog_1    VF3   PropGl_Eth~      2           24 skog_1_VF3_we~     46
## 8 Skog_1    VF3   PropGl_Eth~      2           26 skog_1_VF3_we~     29
## 9 Skog_1    VF3   PropGl_Eth~      2           28 skog_1_VF3_we~     44
## 10 Skog_1   VF3   PropGl_Eth~      2           30 skog_1_VF3_we~     17
## # ... with 119 more rows

liquid_mod_0 <- glmer(no_species ~ liquid * weeks_sampled + (1 | sample_id) + (1 | empty_week |
family = "poisson",
data = liquid_comp)

isSingular(liquid_mod_0)

## [1] FALSE

summary(liquid_mod_0)

## Generalized linear mixed model fit by maximum likelihood (Laplace
## Approximation) [glmerMod]
## Family: poisson  ( log )
## Formula: no_species ~ liquid * weeks_sampled + (1 | sample_id) + (1 |
##   empty_week) + (1 | locality)
## Data: liquid_comp
##
##      AIC      BIC  logLik deviance df.resid
## 1185.2  1205.2   -585.6   1171.2      122
##
```

```

## Scaled residuals:
##      Min     1Q Median     3Q    Max
## -1.41602 -0.19260 -0.00388  0.14588  0.31899
##
## Random effects:
##   Groups      Name      Variance Std.Dev.
##   sample_id  (Intercept) 0.39380  0.6275
##   locality   (Intercept) 0.03340  0.1827
##   empty_week (Intercept) 0.08405  0.2899
## Number of obs: 129, groups: sample_id, 129; locality, 10; empty_week, 7
##
## Fixed effects:
##                               Estimate Std. Error z value Pr(>|z|)
##   (Intercept)                3.59212   0.27922 12.865 <2e-16 ***
##   liquidPropGl_Ethanol       -0.10435   0.34846 -0.299   0.765
##   weeks_sampled              -0.05775   0.09309 -0.620   0.535
##   liquidPropGl_Ethanol:weeks_sampled 0.08326   0.12765  0.652   0.514
##   ---
##   Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##   (Intr) lqPG_E wks_sm
##   lqdPrpGl_Et -0.632
##   weeks_smpld -0.839  0.658
##   lqdPrpG_E:_  0.599 -0.943 -0.701
overdisp_fun <- function(model) {
  rdf <- df.residual(model)
  rp <- residuals(model, type="pearson")
  Pearson.chisq <- sum(rp^2)
  prat <- Pearson.chisq/rdf
  pval <- pchisq(Pearson.chisq, df=rdf, lower.tail=FALSE)
  c(chisq=Pearson.chisq, ratio=prat, rdf=rdf, p=pval)
}
liquid_mod_1 <- glmer(no_species ~ liquid * weeks_sampled + (1 | empty_week) + (1 | locality,
                           family = "poisson",
                           data = liquid_comp)

isSingular(liquid_mod_1)

## [1] FALSE
summary(liquid_mod_1)

## Generalized linear mixed model fit by maximum likelihood (Laplace
## Approximation) [glmerMod]
## Family: poisson  ( log )

```

```

## Formula: no_species ~ liquid * weeks_sampled + (1 | empty_week) + (1 |
##   locality)
## Data: liquid_comp
##
##      AIC      BIC logLik deviance df.resid
## 2647.8 2665.0 -1317.9 2635.8      123
##
## Scaled residuals:
##   Min     1Q Median     3Q    Max
## -6.9510 -2.8243 -0.7343  1.8193 18.4159
##
## Random effects:
## Groups   Name        Variance Std.Dev.
## locality (Intercept) 0.03647  0.1910
## empty_week (Intercept) 0.16544  0.4067
## Number of obs: 129, groups: locality, 10; empty_week, 7
##
## Fixed effects:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  3.87794   0.17732 21.870 < 2e-16 ***
## liquidPropGl_Ethanol       -0.03565   0.08388 -0.425  0.6708
## weeks_sampled                -0.13798   0.02465 -5.597 2.18e-08 ***
## liquidPropGl_Ethanol:weeks_sampled 0.10218   0.03131  3.263  0.0011 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) lqPG_E wks_sm
## lqdPrpGl_Et -0.269
## weeks_smpld -0.337  0.695
## lqdPrpG_E:_  0.260 -0.944 -0.749
overdisp_fun(liquid_mod_1)

##      chisq      ratio      rdf      p
## 2039.12180 16.57823 123.00000 0.00000

overdisp_fun(liquid_mod_0)

##      chisq      ratio      rdf      p
## 13.4765161 0.1104632 122.0000000 1.0000000

liquid_pred <- ggpredict(liquid_mod_0,
  terms = c("liquid", "weeks_sampled"))

plot(liquid_pred) +
  scale_color_nina(name = "Antall uker") +
  ggtitle("") +

```

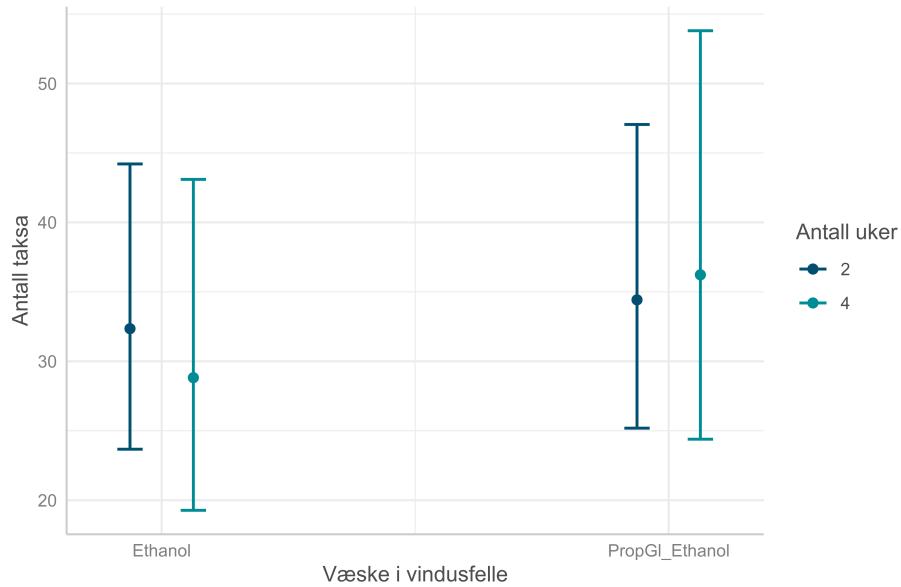


Figure 12: Modelled difference in number of taxa found in window traps, in 2/4 weeks sampling and ethanol/propylen-glycole liquid.

```

ylab("Antall taksa") +
xlab("Væske i vindusfelle")

## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.

```

## Ano-registrations

Here is some simple graphs of the ANO measurements at the locations.

```

ano_arter <- tbl(con,
                  in_schema("ano", "herb_species"))

ano_plots <- tbl(con,
                  in_schema("ano", "survey_points"))

trap_loc <- tbl(con,
                  in_schema("traps", "locations"))

```

```

trap_loc %>%
  left_join(ano_plots,
             by = c("ano_flate_id" = "ano_flate_id")) %>%
  select(-c(geom_punched, geom_inferred)) %>%
  collect() %>%
  select(location, ano_flate_id, global_id) %>%
  distinct() %>%
  print(n = Inf)

## # A tibble: 170 x 3
##   location    ano_flate_id global_id
##   <chr>        <chr>       <chr>
## 1 Semi-nat_1  AN02090     8f1e3337-2266-a776-2ec4-fe1a20f07a91
## 2 Semi-nat_2  AN04425     b2720789-9d0b-5c40-5ca5-f864591c3d5d
## 3 Semi-nat_3  AN03017     dc774f76-2002-3861-4017-cd66781e422c
## 4 Semi-nat_4  AN04145     a823bb2b-610e-35be-7ff3-bd9e263e6a31
## 5 Semi-nat_5  AN02860     5f760aec-1aa6-1e10-7da1-71808f893bf2
## 6 Semi-nat_6  AN09774     3c34ac4e-0788-50d9-083a-b8f304166fb0
## 7 Semi-nat_7  AN04095     53a2a769-81d0-6b51-0067-e50172392a45
## 8 Semi-nat_8  AN06685     6fc6939b-7a55-a074-103d-e7b9b3db09cb
## 9 Semi-nat_9  AN07399     e14b08a6-2b17-351a-4b7b-c9445dbea181
## 10 Semi-nat_10 AN05494    17d0299b-1faa-a090-4403-14b89881572c
## 11 Skog_1      AN01092     <NA>
## 12 Skog_2      AN00958     eacac0ce-c530-4f9b-937f-cf7cdc0a1064
## 13 Skog_2      AN00958     98f01570-3899-437c-bc48-b9355ecd77c3
## 14 Skog_2      AN00958     b17315da-ef8b-4ce3-b9f3-ab9f9a7fddde
## 15 Skog_2      AN00958     d0646269-56ad-453f-9ec3-261763560a69
## 16 Skog_2      AN00958     1ba7dd73-c290-458c-b90a-ec313ee56a24
## 17 Skog_2      AN00958     00151f09-c2ea-4904-bbb8-ee51d338e07f
## 18 Skog_2      AN00958     1d68691b-3139-4691-9267-7562ef6383c9
## 19 Skog_2      AN00958     d96414d1-f1bb-4cbf-b5b8-87fdb42681b1
## 20 Skog_2      AN00958     74367c02-3c1d-4c3f-84d8-9c11a46adb9b
## 21 Skog_2      AN00958     438ae2c4-fa3a-4eb1-a77b-a5139189659a
## 22 Skog_2      AN00958     1eed14ae-2406-4e3d-9c13-b789ac047eee
## 23 Skog_2      AN00958     36d5710d-4f81-474c-8b84-6be1478004a7
## 24 Skog_2      AN00958     b277acf-2984-4e77-939f-56749c3d3185
## 25 Skog_2      AN00958     0aace735-6935-415b-913c-44e2f5f68322
## 26 Skog_2      AN00958     8395488c-816c-4360-9525-18a16aaf2e9e
## 27 Skog_2      AN00958     d625e9be-1839-4dc8-8546-750aae4efd44
## 28 Skog_3      AN00426     50d47680-b9fd-447d-9823-5fb7e2f0fa22
## 29 Skog_3      AN00426     9b6d609d-07d1-4e2f-92ba-14d60f09c2ff
## 30 Skog_3      AN00426     af4d5bf6-0687-47e5-8af8-568f511f8889
## 31 Skog_3      AN00426     4f122810-a082-4449-9c0c-6b5e60bf56f3
## 32 Skog_3      AN00426     7f1f588c-3d05-40ac-a706-8396530c4c9f
## 33 Skog_3      AN00426     bf617829-9a06-426d-b935-63570717a28c

```

##	34	Skog_3	AN00426	3a8edc11-af8d-4537-8da8-c902067c8d83
##	35	Skog_3	AN00426	bb8d62af-1f1c-4143-8e35-ade2259a5b52
##	36	Skog_3	AN00426	ba3f3fee-0612-4e4e-a076-990e792e2494
##	37	Skog_3	AN00426	4a93708b-6cd0-4a97-97bc-217482cb74d8
##	38	Skog_3	AN00426	82c85b12-b13c-4316-a0c3-9c6f2f54957a
##	39	Skog_3	AN00426	bb957353-9dac-4167-a3ea-41e34367571d
##	40	Skog_3	AN00426	d8f16f8b-01ae-430e-b5a7-163ee8f1cee2
##	41	Skog_3	AN00426	f6154d42-2f83-4464-a360-faac441a917a
##	42	Skog_3	AN00426	a4c5252f-92f2-4784-8a26-b9aff0f7dba8
##	43	Skog_3	AN00426	10663364-b8f5-48e2-921d-6b3fafd0ca77
##	44	Skog_3	AN00426	bad8d4b5-199f-4b07-af45-7cbfdfffc99fa
##	45	Skog_3	AN00426	44fa14e7-6e64-40bc-98f8-d1bcce55895d
##	46	Skog_4	AN00574	6fb273f-d59b-4fc4-94f6-e6426f74d95a
##	47	Skog_4	AN00574	86319924-3495-4572-b01c-86d0ebab2417
##	48	Skog_4	AN00574	3adaf83f-d835-4a26-ab73-6ebac96af38b
##	49	Skog_4	AN00574	ebc51dc9-3842-4a20-8360-3a60c66abc5a
##	50	Skog_4	AN00574	1c2f25d0-2d5e-4aab-9724-cf84a125ff4a
##	51	Skog_4	AN00574	fa7381bd-4323-48b6-b3c4-0acbacc553c77
##	52	Skog_4	AN00574	373ac331-911c-4359-8b95-1623d19aa6dc
##	53	Skog_4	AN00574	0d3022e5-182d-4838-87c5-ad800511fcdd
##	54	Skog_4	AN00574	1427d114-6db8-4078-a673-82567b270251
##	55	Skog_4	AN00574	7fca1212-c746-45cc-89c4-ff68623bc25c
##	56	Skog_4	AN00574	9618245a-9c95-4807-a6cb-0c5dabe567db
##	57	Skog_4	AN00574	1efd2bafe-fc92-4d32-8ac2-d63e11221f8c
##	58	Skog_4	AN00574	c6cb52dd-fa15-4679-969e-aac737feb040
##	59	Skog_4	AN00574	87535e15-9fb9-4966-8e25-6af45c4b07b
##	60	Skog_4	AN00574	eeb6f92c-b48b-4e9f-b47d-63e636cbccae
##	61	Skog_4	AN00574	f32db2c5-9446-48e2-8e55-d935af463f53
##	62	Skog_4	AN00574	8a281e97-f73e-4186-9253-855d6d0001d9
##	63	Skog_5	AN00800	cc27776b-2fdb-49ac-9a84-a0ada7c072c8
##	64	Skog_5	AN00800	b43fd424-9a27-42b5-97ce-31e48a8da2a8
##	65	Skog_5	AN00800	8f11d82c-327f-4417-9650-50d139b68acc
##	66	Skog_5	AN00800	5e58b79d-783e-49d2-8105-115d5ad3c97f
##	67	Skog_5	AN00800	8f472294-92cd-4c86-b563-f4023633646b
##	68	Skog_5	AN00800	6adeaa18-9172-4cfc-bfd4-4bf9d2d2d40f
##	69	Skog_5	AN00800	d8f5eb87-9fa1-41aa-8c7f-f4f6cabd784a
##	70	Skog_5	AN00800	558e0857-02b1-4e26-80a7-d7874af78fb7
##	71	Skog_5	AN00800	2e751260-2c79-4223-85f3-3b389b5dd623
##	72	Skog_5	AN00800	eae98d16-8b5e-40a4-afa7-93fb9f4493c2
##	73	Skog_5	AN00800	4b617538-df95-4ed6-87a3-45c324c765c7
##	74	Skog_5	AN00800	1b290a93-a182-43d4-aa5a-7a1c815fa4e3
##	75	Skog_5	AN00800	48de301a-6814-4ad1-ab8f-825b2f749929
##	76	Skog_5	AN00800	14d75cf1-b06c-4371-94fd-5c988771daf1
##	77	Skog_5	AN00800	4d4b70c8-5856-4348-a0fd-008927488d1b
##	78	Skog_5	AN00800	51ff49da-2371-4abc-b5e4-e30313de6108
##	79	Skog_5	AN00800	f9953dce-3264-43f0-a59e-ed6045ca3626

## 80 Skog_5	AN00800	6943555f-ae71-4210-8ebe-c1f6d628c415
## 81 Skog_6	AN00856	ad910f07-604e-41c6-b0d9-9cd8a31b2185
## 82 Skog_6	AN00856	2b4e63fb-319f-446d-9637-a0dc05a2604e
## 83 Skog_6	AN00856	9bb6eef8-8d05-435d-b1e0-8d8ab84681a7
## 84 Skog_6	AN00856	c4e7bdb1-6b6e-4dbb-ad46-e1dc64dccd4e
## 85 Skog_6	AN00856	6939e70b-6844-47f8-bcef-61c0b30bde87
## 86 Skog_6	AN00856	43fb5f6b-6d6a-44d8-8f2a-3dea56177674
## 87 Skog_6	AN00856	58ac69c7-e288-4f26-9bf5-0232183c4adf
## 88 Skog_6	AN00856	8eb7f451-82af-4c68-aeaf-770c2ee40f1a
## 89 Skog_6	AN00856	ced9ff6f-46e6-4225-b6b5-0fac3a4eb26c
## 90 Skog_6	AN00856	c34b72b3-2c0d-4a90-aab5-3469b685efe3
## 91 Skog_6	AN00856	094d7455-5849-45bb-818e-13d6d7a565aa
## 92 Skog_6	AN00856	0b345076-5109-40f6-9bd2-99324c1f541d
## 93 Skog_6	AN00856	cd32c1ad-efcb-4688-9509-b13c1eadb9e8
## 94 Skog_6	AN00856	a011438c-33a6-4ad7-aebb-1b64a34779a2
## 95 Skog_6	AN00856	034e1f5e-c02c-4465-849a-2937a59afaa0
## 96 Skog_6	AN00856	ad30cab3-8320-4905-9120-22915ac57e4c
## 97 Skog_6	AN00856	81e3fe2b-24b2-4208-aad7-614da78aed48
## 98 Skog_6	AN00856	180007e6-37f7-464b-804c-4a01f24cc680
## 99 Skog_7	AN00534	4b0f9cc3-4529-437e-b95b-2def395159b9
## 100 Skog_7	AN00534	c2a914ca-75ca-4a2b-88d3-c7eefdb86bea
## 101 Skog_7	AN00534	0ff427a1-1ec5-43b1-bb8d-f283f07b56b6
## 102 Skog_7	AN00534	ad708040-36e6-46c9-94df-ad1578bc3018
## 103 Skog_7	AN00534	d9a35b41-851a-44cf-9f2b-183a981f19cc
## 104 Skog_7	AN00534	b69221c3-6525-478c-878d-a4ae79c233d7
## 105 Skog_7	AN00534	e094d311-57fd-40f7-add9-c0a8f576c44e
## 106 Skog_7	AN00534	a8af17f8-ea1a-41cf-bc77-e548eb22a575
## 107 Skog_7	AN00534	ac84b604-b999-40ee-af2c-c3e3757c2b01
## 108 Skog_7	AN00534	f1369211-4763-414b-9704-3ab816340ef9
## 109 Skog_7	AN00534	8cc22e10-b412-4788-8c15-2015578b7e9e
## 110 Skog_7	AN00534	3a11ab54-aa71-431c-b7ec-a0d1f25c17f6
## 111 Skog_7	AN00534	b6fd6696-e5fc-4016-a8d0-a190d2825922
## 112 Skog_7	AN00534	a510b524-b803-472c-b311-1b5db9807595
## 113 Skog_7	AN00534	19fb1513-ef46-4a32-9f76-2cbdaccb6999
## 114 Skog_7	AN00534	6828fd28-2a82-4098-8244-354996b65c63
## 115 Skog_7	AN00534	fd53364a-f6b4-433f-b782-4f98f35754c7
## 116 Skog_7	AN00534	039d8eb8-779a-4c19-bbb1-ec62544c21cb
## 117 Skog_8	AN00051	397ab2fc-7719-4235-8fa9-f4937591f991
## 118 Skog_8	AN00051	d76b3990-0d7f-4915-8ad1-85b634536148
## 119 Skog_8	AN00051	6c249d32-46f1-4250-b060-db1d9e8de3ed
## 120 Skog_8	AN00051	3d32b4a0-3e4d-4318-ad24-4e84894aca08
## 121 Skog_8	AN00051	9327696f-407e-4938-b418-d6989d530a26
## 122 Skog_8	AN00051	8ae13c78-2821-48f7-8eae-8f5d271aa950
## 123 Skog_8	AN00051	1b69394d-c000-4a35-b403-33b64c9deb69
## 124 Skog_8	AN00051	4a06f6f3-d677-45ed-98b9-236c6972763d
## 125 Skog_8	AN00051	0c168294-1f84-47f5-a8a6-c06785bf46a0

## 126 Skog_8	AN00051	93605b59-a4e8-4d3e-8ff1-a55e19b67b8c
## 127 Skog_8	AN00051	829b6c69-ccbe-47ab-a9aa-0aa7ec31ef81
## 128 Skog_8	AN00051	4e01548f-5311-445b-be8c-adc57f98eb39
## 129 Skog_8	AN00051	dc8d0099-aa14-4f05-8869-afad40591484
## 130 Skog_8	AN00051	0a2250f4-16a0-48d9-85e0-1e8e8f7b3ec0
## 131 Skog_8	AN00051	a9417c27-0554-436e-94ca-4a1dfe4d7a8a
## 132 Skog_8	AN00051	324ba37b-160e-4413-ad23-c697fba387b7
## 133 Skog_8	AN00051	2ea4d8ff-06b2-4ea4-9525-810969d7beed
## 134 Skog_8	AN00051	d6e6b5e2-d9f8-49aa-b014-2796acd67417
## 135 Skog_9	AN00914	6fd9b9d2-7893-473a-8b67-a2ba398414a7
## 136 Skog_9	AN00914	f5b9b0fe-7b48-42e9-88a1-a8b3c04f488a
## 137 Skog_9	AN00914	8f15d6e3-97c1-458d-8b42-4d12362fdbfa
## 138 Skog_9	AN00914	95838ef7-551b-48d5-af85-15c0df6621de
## 139 Skog_9	AN00914	5c42ec16-88df-4323-80e5-571a980326cf
## 140 Skog_9	AN00914	98ee5314-56ec-4d86-94ae-bcab139e5249
## 141 Skog_9	AN00914	bac49f20-4048-4e00-a9a4-d07153e04bea
## 142 Skog_9	AN00914	d9c1aafb-41a7-469e-8454-ce9b8423962a
## 143 Skog_9	AN00914	a5868ed4-cd56-4367-a7c0-5c32ce3ad416
## 144 Skog_9	AN00914	d41ed974-ff53-4a64-b0cc-326d990374a8
## 145 Skog_9	AN00914	39f6b55f-ed5f-4389-8d8d-602b3baf1be8
## 146 Skog_9	AN00914	106ce33d-3612-4577-bd37-a09331a2044d
## 147 Skog_9	AN00914	f59861bf-ad43-4c2c-a032-f44a15ed8534
## 148 Skog_9	AN00914	9660d2c0-099f-4091-8ce2-d5debfa10c73
## 149 Skog_9	AN00914	737f80e4-c9db-4c3f-ae0a-23699803a968
## 150 Skog_9	AN00914	2e07c31a-4f0f-44df-b103-e7982bf09f92
## 151 Skog_9	AN00914	82f4e1bc-4746-4823-a1eb-4cdc83ccee43
## 152 Skog_9	AN00914	d962d6ac-be4e-4046-893d-9b94779e497c
## 153 Skog_10	AN00588	694fab07-c095-49ac-b64a-609a21ccc417
## 154 Skog_10	AN00588	6e0eba4e-0414-4e12-b0cc-ecc4939bb4cb
## 155 Skog_10	AN00588	ac660d80-ed6a-4d2a-9dc7-b7ac78ecc0fd
## 156 Skog_10	AN00588	9702fad8-81de-4a6f-8e69-fda7ecd517a1
## 157 Skog_10	AN00588	a299825e-dcbb-4af8-95a9-2ccd260b254f
## 158 Skog_10	AN00588	bb0c08db-1a6a-4cad-a766-6bedaed5bc04
## 159 Skog_10	AN00588	2a4e885f-67fc-47da-ad3e-e32c8e72257f
## 160 Skog_10	AN00588	56800450-c3c7-4df6-a0f4-e71d69a36e15
## 161 Skog_10	AN00588	f43ebd65-a5ef-4d5b-9ab9-42bf205b2bc0
## 162 Skog_10	AN00588	21b60b22-89ed-46df-ae03-0b094defaf63
## 163 Skog_10	AN00588	007a3d44-309c-4863-b4bc-431809da3da3
## 164 Skog_10	AN00588	21d064e7-11ca-45f6-9136-681ef02d14c0
## 165 Skog_10	AN00588	47bdc52d-d8a5-43cd-8612-d41024c8db70
## 166 Skog_10	AN00588	65c73bf7-ea26-44b6-9749-326a9c54fb0a
## 167 Skog_10	AN00588	a83d70e8-29b1-4636-b503-22651f6c1b7a
## 168 Skog_10	AN00588	d07bc0e7-5aa1-452a-8b63-f2836ad71c7d
## 169 Skog_10	AN00588	fafcf9d24-81f3-40cc-8ab5-4477e63b0844
## 170 Skog_10	AN00588	07b69d68-5258-4e98-ab1f-cb42dde616a

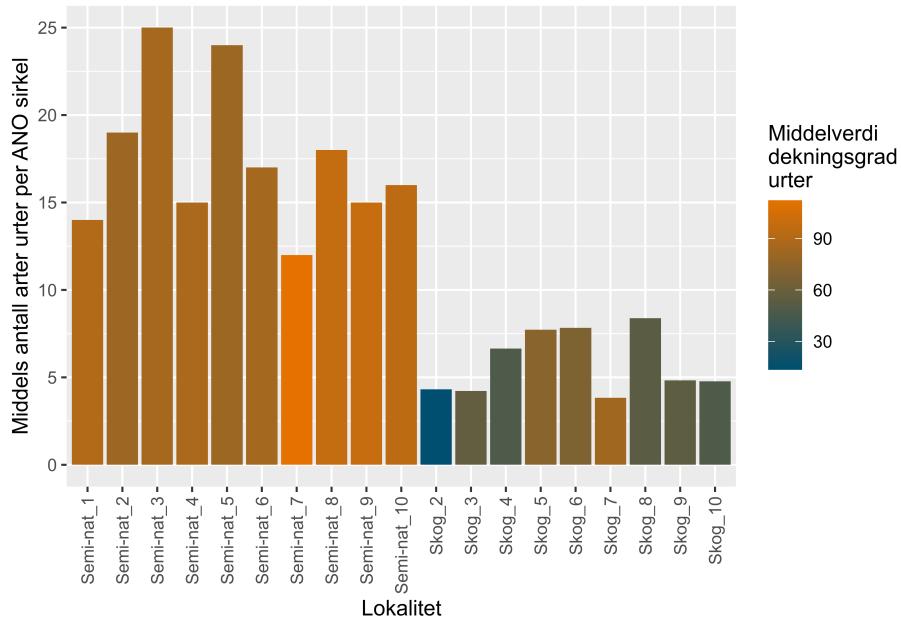
```

ano_herb_spec <- trap_loc %>%
  left_join(ano_plots,
             by = c("ano_flate_id" = "ano_flate_id")) %>%
  left_join(ano_arter,
             by = c("global_id" = "parent_global_id")) %>%
  select(-c(geom_punched, geom_inferred)) %>%
  collect()

ano_agg <- ano_herb_spec %>%
  group_by(global_id.x) %>%
  summarize(location = first(location),
            no_spec = n_distinct(navn),
            cover = sum(dekning_prc, na.rm = T)) %>%
  group_by(location) %>%
  summarise(mean_no_spec = mean(no_spec),
            mean_cover = mean(cover)) %>%
  collect() %>%
  mutate(location = factor(location, levels = c(paste0("Semi-nat_", 1:10), paste0("Skog_", 1:10)),
                           filter(location != "Skog_1"))

## `summarise()` ungrouping output (override with ` `.groups` argument)
## `summarise()` ungrouping output (override with ` `.groups` argument)
ano_agg %>%
  ggplot(.) +
  geom_bar(aes(x = location, y = mean_no_spec, fill = mean_cover), stat = "identity") +
  scale_fill_nina(name = "Middelverdi\\ndekningsgrad\\nurter",
                  discrete = F,
                  palette = "darkblue-orange") +
  ylab("Middels antall arter urter per ANO sirkel") +
  xlab("Lokalitet") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

```



## Forest-registrations

Some simple graphs of the forest registrations.

```
tree_reg <- tbl(con,
                  in_schema("landsskog", "survey_data"))

tree_plots <- tbl(con,
                  in_schema("landsskog", "survey_points"))

trap_loc <- tbl(con,
                  in_schema("traps", "locations"))

tree_spec <- tbl(con,
                  in_schema("landsskog", "tree_species"))

tree_spec <- tree_reg %>%
  left_join(tree_plots,
             by = c("parent_id" = "id")) %>%
  left_join(tree_spec,
             by = c("treslag_kode" = "treslag_kode")) %>%
  left_join(trap_loc,
             by = c("ano_flate_id" = "ano_flate_id"))
```

Which tree got age determined?

```
tree_spec %>%
  select(location, treslag, boniteringstre) %>%
  arrange(location) %>%
  print(n = Inf)

## # Source:     lazy query [?? x 3]
## # Database:   postgres
## #   [jens.astrom@ninradardata01.nina.no:5432/insect_monitoring]
## # Ordered by: location
##   location treslag  boniteringstre
##   <chr>     <chr>    <lgl>
## 1 Skog_1     Furu     FALSE
## 2 Skog_1     Furu     FALSE
## 3 Skog_1     Furu     TRUE
## 4 Skog_1     Dunbjerk TRUE
## 5 Skog_1     Dunbjerk FALSE
## 6 Skog_1     Dunbjerk FALSE
## 7 Skog_1     Furu     FALSE
## 8 Skog_1     Furu     FALSE
## 9 Skog_10    Gran    TRUE
## 10 Skog_10   Gran    FALSE
## 11 Skog_10   Gran    FALSE
## 12 Skog_10   Gran    TRUE
## 13 Skog_10   Gran    FALSE
## 14 Skog_10   Gran    FALSE
## 15 Skog_10   Furu    FALSE
## 16 Skog_2    Furu    FALSE
## 17 Skog_2    Furu    FALSE
## 18 Skog_2    Furu    FALSE
## 19 Skog_2    Furu    FALSE
## 20 Skog_2    Furu    FALSE
## 21 Skog_2    Furu    FALSE
## 22 Skog_2    Furu    FALSE
## 23 Skog_2    Gran    FALSE
## 24 Skog_2    Furu    FALSE
## 25 Skog_2    Osp     FALSE
## 26 Skog_2    Furu    FALSE
## 27 Skog_2    Furu    FALSE
## 28 Skog_2    Furu    TRUE
## 29 Skog_2    Furu    TRUE
## 30 Skog_3    Gran    FALSE
## 31 Skog_3    Furu    FALSE
## 32 Skog_3    Gran    FALSE
## 33 Skog_3    Gran    FALSE
```

```

## 34 Skog_3 Furu FALSE
## 35 Skog_3 Furu FALSE
## 36 Skog_3 Gran FALSE
## 37 Skog_3 Gran FALSE
## 38 Skog_3 Gran FALSE
## 39 Skog_3 Furu TRUE
## 40 Skog_3 Furu FALSE
## 41 Skog_3 Furu TRUE
## 42 Skog_3 Furu FALSE
## 43 Skog_3 Furu FALSE
## 44 Skog_3 Furu FALSE
## 45 Skog_3 Gran FALSE
## 46 Skog_3 Gran FALSE
## 47 Skog_4 Gran FALSE
## 48 Skog_4 Furu TRUE
## 49 Skog_4 Dunbjerk FALSE
## 50 Skog_4 Dunbjerk FALSE
## 51 Skog_4 Dunbjerk FALSE
## 52 Skog_4 Furu TRUE
## 53 Skog_4 Dunbjerk FALSE
## 54 Skog_4 Dunbjerk FALSE
## 55 Skog_4 Furu FALSE
## 56 Skog_5 Dunbjerk FALSE
## 57 Skog_5 Dunbjerk FALSE
## 58 Skog_5 Dunbjerk FALSE
## 59 Skog_5 Dunbjerk FALSE
## 60 Skog_5 Dunbjerk FALSE
## 61 Skog_5 Dunbjerk FALSE
## 62 Skog_5 Dunbjerk FALSE
## 63 Skog_5 Dunbjerk FALSE
## 64 Skog_5 Dunbjerk FALSE
## 65 Skog_5 Dunbjerk FALSE
## 66 Skog_5 Dunbjerk FALSE
## 67 Skog_5 Dunbjerk FALSE
## 68 Skog_5 Dunbjerk FALSE
## 69 Skog_5 Dunbjerk FALSE
## 70 Skog_5 Dunbjerk FALSE
## 71 Skog_5 Dunbjerk FALSE
## 72 Skog_5 Dunbjerk FALSE
## 73 Skog_5 Dunbjerk FALSE
## 74 Skog_5 Dunbjerk FALSE
## 75 Skog_5 Dunbjerk FALSE
## 76 Skog_5 Dunbjerk FALSE
## 77 Skog_5 Dunbjerk FALSE
## 78 Skog_5 Dunbjerk FALSE
## 79 Skog_5 Dunbjerk FALSE

```

```
## 80 Skog_5    Dunbjerk FALSE
## 81 Skog_5    Dunbjerk FALSE
## 82 Skog_5    Dunbjerk TRUE
## 83 Skog_5    Dunbjerk FALSE
## 84 Skog_5    Dunbjerk FALSE
## 85 Skog_5    Dunbjerk TRUE
## 86 Skog_5    Dunbjerk FALSE
## 87 Skog_5    Dunbjerk FALSE
## 88 Skog_6    Dunbjerk FALSE
## 89 Skog_6    Dunbjerk FALSE
## 90 Skog_6    Dunbjerk FALSE
## 91 Skog_6    Dunbjerk FALSE
## 92 Skog_6    Dunbjerk FALSE
## 93 Skog_6    Dunbjerk FALSE
## 94 Skog_6    Dunbjerk FALSE
## 95 Skog_6    Dunbjerk FALSE
## 96 Skog_6    Dunbjerk FALSE
## 97 Skog_6    Dunbjerk FALSE
## 98 Skog_6    Dunbjerk FALSE
## 99 Skog_6    Dunbjerk FALSE
## 100 Skog_6   Dunbjerk FALSE
## 101 Skog_6   Dunbjerk FALSE
## 102 Skog_6   Dunbjerk FALSE
## 103 Skog_6   Dunbjerk FALSE
## 104 Skog_6   Dunbjerk FALSE
## 105 Skog_6   Dunbjerk FALSE
## 106 Skog_6   Dunbjerk TRUE
## 107 Skog_6   Dunbjerk FALSE
## 108 Skog_6   Dunbjerk FALSE
## 109 Skog_6   Dunbjerk FALSE
## 110 Skog_6   Dunbjerk TRUE
## 111 Skog_6   Dunbjerk FALSE
## 112 Skog_6   Dunbjerk FALSE
## 113 Skog_6   Dunbjerk FALSE
## 114 Skog_6   Dunbjerk FALSE
## 115 Skog_6   Dunbjerk FALSE
## 116 Skog_7   Furu    FALSE
## 117 Skog_7   Furu    FALSE
## 118 Skog_7   Furu    FALSE
## 119 Skog_7   Furu    FALSE
## 120 Skog_7   Furu    FALSE
## 121 Skog_7   Furu    FALSE
## 122 Skog_7   Furu    FALSE
## 123 Skog_7   Furu    FALSE
## 124 Skog_7   Furu    FALSE
## 125 Skog_7   Furu    FALSE
```

```
## 126 Skog_7   Furu   FALSE
## 127 Skog_7   Furu   FALSE
## 128 Skog_7   Furu   FALSE
## 129 Skog_7   Furu   TRUE
## 130 Skog_7   Gran   TRUE
## 131 Skog_7   Furu   FALSE
## 132 Skog_7   Furu   FALSE
## 133 Skog_7   Furu   FALSE
## 134 Skog_7   Furu   FALSE
## 135 Skog_7   Furu   FALSE
## 136 Skog_7   Furu   FALSE
## 137 Skog_8   Gran   FALSE
## 138 Skog_8   Dunbjerk FALSE
## 139 Skog_8   Dunbjerk FALSE
## 140 Skog_8   Dunbjerk FALSE
## 141 Skog_8   Furu   TRUE
## 142 Skog_8   Dunbjerk FALSE
## 143 Skog_8   Dunbjerk FALSE
## 144 Skog_8   Dunbjerk FALSE
## 145 Skog_8   Dunbjerk FALSE
## 146 Skog_8   Gran   FALSE
## 147 Skog_8   Gran   TRUE
## 148 Skog_8   Dunbjerk FALSE
## 149 Skog_8   Gran   FALSE
## 150 Skog_8   Dunbjerk FALSE
## 151 Skog_8   Dunbjerk FALSE
## 152 Skog_8   Dunbjerk FALSE
## 153 Skog_8   Dunbjerk FALSE
## 154 Skog_8   Dunbjerk FALSE
## 155 Skog_8   Dunbjerk FALSE
## 156 Skog_8   Gran   FALSE
## 157 Skog_8   Dunbjerk FALSE
## 158 Skog_9   Gran   FALSE
## 159 Skog_9   Gran   TRUE
## 160 Skog_9   Gran   FALSE
## 161 Skog_9   Gran   FALSE
## 162 Skog_9   Gran   FALSE
## 163 Skog_9   Gran   FALSE
## 164 Skog_9   Gran   FALSE
## 165 Skog_9   Gran   FALSE
## 166 Skog_9   Gran   FALSE
## 167 Skog_9   Gran   FALSE
## 168 Skog_9   Gran   FALSE
## 169 Skog_9   Gran   FALSE
## 170 Skog_9   Gran   FALSE
## 171 Skog_9   Gran   FALSE
```

```

## 172 Skog_9   Gran   FALSE
## 173 Skog_9   Gran   FALSE
## 174 Skog_9   Gran   FALSE
## 175 Skog_9   Gran   FALSE
## 176 Skog_9   Gran   TRUE
## 177 Skog_9   Gran   FALSE
## 178 Skog_9   Gran   FALSE
## 179 Skog_9   Gran   FALSE
## 180 Skog_9   Gran   FALSE
## 181 Skog_9   Gran   FALSE

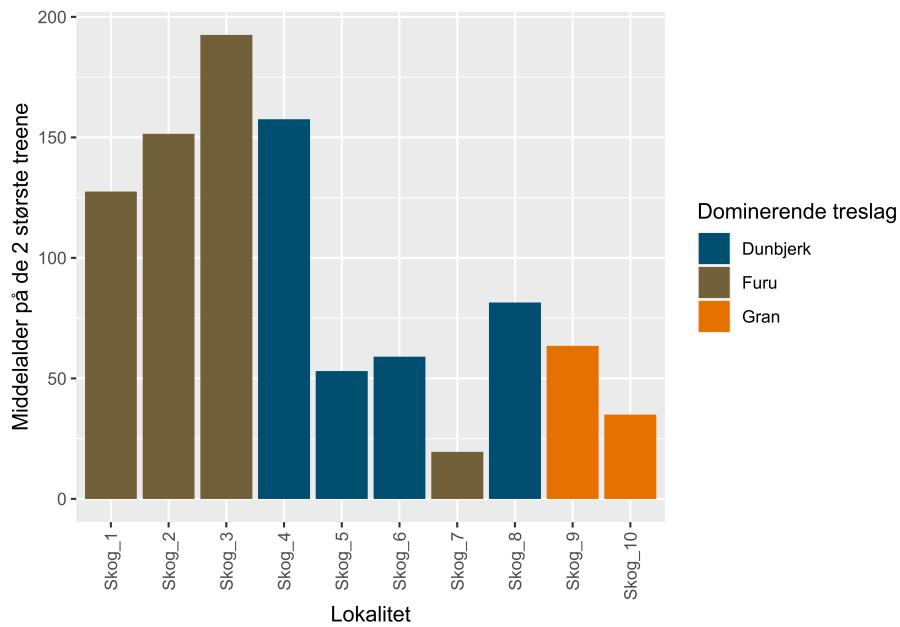
dom_tree <- tree_spec %>%
  group_by(location,
           treslag) %>%
  summarise(no_trees = n()) %>%
  collect() %>%
  arrange(location,
          desc(no_trees)) %>%
  slice_min(1) %>%
  select(dom_tree = treslag,
         -no_trees)

## Adding missing grouping variables: `location`

tree_agg <- tree_spec %>%
  collect() %>%
  left_join(dom_tree,
            by = c("location" = "location")) %>%
  group_by(location, dom_tree) %>%
  summarize(avg_age = mean(alder, na.rm = T)) %>%
  collect() %>%
  mutate(location = factor(location, levels = paste0("Skog_", 1:10)))

## `summarise()` regrouping output by 'location' (override with `groups` argument)
tree_agg %>%
  ggplot(.) +
  geom_bar(aes(x = location, y = avg_age, fill = dom_tree), stat = "identity") +
  scale_fill_nina(name = "Dominerende treslag",
                  discrete = T,
                  palette = "darkblue-orange") +
  ylab("Middelalder på de 2 største treene") +
  xlab("Lokalitet") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

```

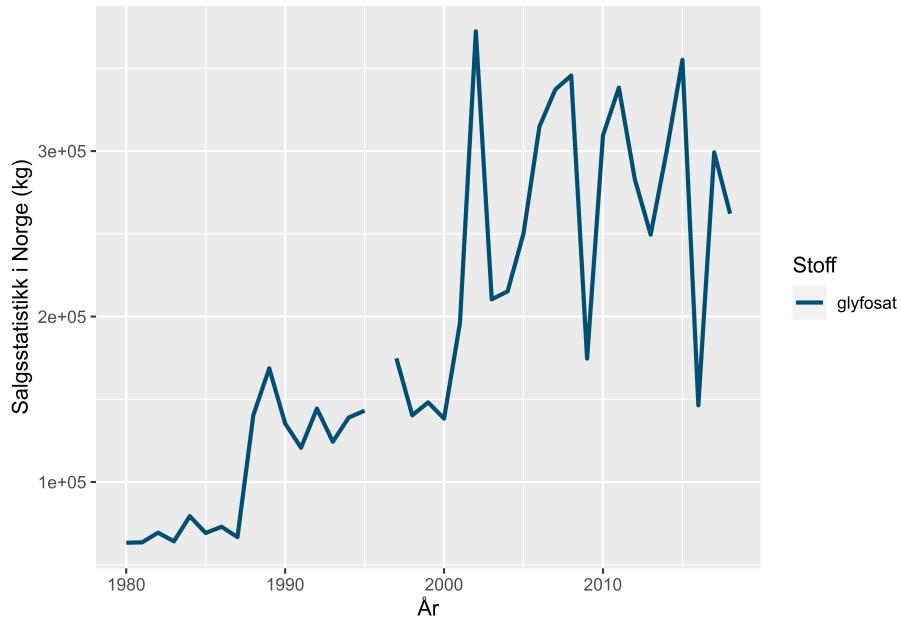


## Glyphosate figure

A simple figure of the sales statistics of glyphosate in Norway

```
herb_pest_data <- read_csv("../Data/Plantevern/glyphosate_neonic.csv")
```

```
##  
## -- Column specification -----  
## cols(  
##   year = col_double(),  
##   type = col_character(),  
##   amount = col_double()  
## )  
herb_pest_data %>%  
  filter(type == "glyfosat") %>%  
  ggplot(.) +  
  geom_line(aes(x = year, y = amount, color = type), lwd = 1) +  
  scale_color_nina(name = "Stoff") +  
  ylab("Salgsstatistikk i Norge (kg)") +  
  xlab("År")
```



```

herb_pest_data %>%
  filter(type == "imidakloprid") %>%
  ggplot(.) +
  geom_line(aes(x = year, y = amount, color = type), lwd = 1) +
  scale_color_nina(name = "Stoff") +
  ylab("Salgsstatistikk i Norge (kg)") +
  xlab("År")

## Warning: Removed 1 row(s) containing missing values (geom_path).

```

